GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Network Monitoring and Threat Detection In-Depth (Security 503)"
at http://www.giac.org/registration/gcia

Open Source Host Based Intrusion Detections System (OHIDS)

GIAC (GCIA) Gold Certification

Author: Tom Webb,Tcw3bb@gmail.com
Advisor: MANUEL HUMBERTO SANTANDER PELAEZ

Accepted: 8th June 2013

Abstract

Responding to incidents in an efficient manner is critical for all CIRTS. This paper presents a new open source tool for the enterprise. With this tool, responders will be able to detect incidents using aggregated data collected from hosts and applying anomaly detection. OHIDS includes a sensitive data finder to allow appropriate escalation of the incident. This software can be utilized in a proactive manner by removing SSNs and credit card data before incidents occur or by detecting unauthorized software running.

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

1. Overview

Detecting and analyzing intrusion based solely on network traffic gives you an incomplete picture, especially if you are lacking full packet captures or if you have a large number of mobile users who do not always use your Internet connection. This is a common struggle with most Computer Incident Response Teams (CIRTs) and they need a fast way to determine if the system is compromised no matter the location. A host based IDS (HIDS) for Windows that feeds into a central database and uses the power of base lining and comparison can meet these needs. OHIDS collects running processes and other key detection mechanisms, along with a sensitive data scanner. By using this HIDS in conjunction with network based alerting, it should allow CIRTs to assess if not only the attack was successful, but also the impact based on what data is on the system.

According to Caswell, Beale and Baker (2007), Snort introduction to the world was on December 22, 1988 (Chapter 2, what is Snort). Network IDS can detect many different malicious activities including exploits, port scans, non-compliant protocols and brute force attacks. Network IDS is also a great way to detect malware infection because most malware needs a way to communicate back to the attacker. This method of analysis is called extrusion detection (Bejtlich, 2006, p. 4).

One of the main draw backs of traditional network analysis is that malware can easily encrypt, obfuscate or purposefully mangle its communication over the network. In a resent GCIA Gold, "Beating the IPS", Michael Dyrmose (2013) discusses how easy it is to bypass IPS technologies (p. 59). To combat just encrypted traffic, you will need to force all traffic to be decrypt before it enters or exits your network. In many environments, this is not possible due to privacy concerns, which enables malware to sneak past with ease.

How do you detect malware when we do not have a signature? How do you detect a laptop compromised at home? Base lining the network and hosts is the best defense against this, but is a difficult thing to accomplish in large distributed environment.

What we are missing is determining the impact of the infection on the system. Depending on your environment, you may be able to segment departments into different subnets. This would give you some indication that a system in the human resource

Tom Webb, tcw3bb@gmail.com

© 2013 The SANS Institute Author retains full rights.

department, but you still do not know if the individuals computer has personally identifiable information (PII) or how much of it.

To fill this gap an open source HIDS will help determine anomalous activity on the system. Some of the more common features in HIDS include log monitoring, process monitoring and network traffic analysis. These features can be used to help detect basic malware, intelligent advisories or even configuration management. Once detected from the network, the information gathered from forensics analysis can be used to query OHIDS to find additional infections.

According to Innella (2001), Haystack Labs released a product name Stalker in 1989 (A Brief History, para. 5). This appears to be the first commercial version of a HIDS. A presentation from Steve Smaha (1996), the president Haystack Labs, states the patent pending technology includes gather data from "Processing system audit trail records, system log file data, and system security state data" (slide 7). While OHIDS is collecting a lot more information, the basic ideas from the Stalker software still apply today.

OSSEC is a great open source project that does log analysis and can detect changes to the registry, file system changes and many other things. The main difference between OSSEC and OHIDS is that you cannot easily use OSSEC as an incident response tool to collect live data from a large number of hosts.

2. Introduction to OHIDS

The first version of OHIDS started in 2007 and it was a batch script to collect basic information. An intern, Maryam Jafari in 2009, wrote the second generation in VB script. This version added a few additional items to collect and the output was dumped to a network share. Another intern, Vipul Gupta in 2010, ported the third version to VB.net. Additional functionality of hashing of processes and loaded modules was added. In the fourth version, output was converted to CSV and imported into a database. This, the fifth version, has the capability to connect to the database making it the first true version where others can easily deploy it in their own environment.

© 2013 The SANS Institute Author retains full rights.

2.1. Architecture

The OHIDS client is written in VB.net using the 4.0 framework. It will collect the following critical system stats: running processes, loaded DLLs, network connections, firewall rule set, services, startup items and scheduled tasks. Each of these stats can be used to determine what is currently running on the system and what functionality the application may have. These items will allow us to differentiate what is normal and what is abnormal for systems on the network.

Each system will connect back to a database server, currently on a daily interval, were the data will be processed and correlated. Additionally, every seven days, the OHIDS client will run a sensitive data scan, using a modified version of Find_SSN(Find_SSN, 2009), on pre-defined directories and insert the results into a table on the database server. Find_SSN is a python script compiled for windows, which scans for social security numbers and credit card numbers. The version packaged with the system is modified to reset the access times to any file that it reads. This prevents stomping on any potential forensics evidence in case a full investigation is required.

2.2. Known Weaknesses

The VB.net client collects data from the system using the Windows API calls and built-in windows utilities. While this technique is convenient and allows for fast development, malware can bypass this method of detection by hooking API calls, injecting DLL's or using a rootkit to manipulate the data OHIDS is collecting. Lots of malware still does not use these techniques and OHIDS is able to find them.

To detect hooks, rootkits or injected DLLs you can use a malware scanner like Malwarebytes Anti-rootkit or GMER that looks at program anomalies. My preferred method for detecting this kind of malware is memory analysis using Mandiants' Redline to detect anomalies by directly accessing windows memory.

Tom Webb, tcw3bb@gmail.com

© 2013 The SANS Institute Author retains full rights.

2.3. Use Cases

2.3.1. Sensitive Data, Malware Type, and Incident Classification

In most enterprise environments, network IDS detects infections many times an hour. Each of these infections must be prioritized and categorized appropriately to use the incident responders' efficiently.

According to Cichonski, Millar, Grance, and Scarfone (2012) your incident response policy should include your expected response time based on assets criticality and threat level of the malware (p. 32). If you are lucky enough to have an enterprise data loss prevention software deployed, then you should be leveraging this information to set response priority. If not, you will need to rely on the helpdesk or have your responder contact the user to determine what information they use on a daily basis. Based on my experience, this method has proven to be very unreliable. Users generally do not want to "Get into Trouble" if they use sensitive data and therefore downplay the data they access. Other users do not want to lose productivity time and have been less truthful. Some users have a hard time actually determining what sensitive data means, even after several discussions. With OHIDS, you have data pre-populated in a database to query. This can be a huge time saver and enable your organization to have consistent data to reference.

2.3.2. Detecting Unknown Malware

Malware is a difficult problem that IT professionals have been fighting for a long time, but after doing incident response for a while one thing seem to be the same. Most malware wants to have a persistence mechanism to insure the system is infected as long as possible. This seems to be one of the fastest and easiest ways of detecting most unknown malware, especially if you have a large list of data to compare. While this method will not catch rootkits or other techniques, it should work well for run-of-the-mill infections.

Finding unknown malware requires a basic understanding of malware and analytics to identify places where it is hiding. The following is a list of simple but powerful techniques that will be used for detecting malware.

1. Least occurrence analysis of registry settings, running processes, and loaded DLL's when compared to all known PC in your infrastructure.

© 2013 The SANS Institute Author retains full rights.

2. Look for common locations where malware hides
3. Compare daily system changes in the registry, running processes and services to help focus analysis.

Least occurrence analysis can be used to determine unique settings or process running on systems. With a large number of systems in the database, most systems will have the same applications running. Individual computers running a unique process stand out. Many version of malware still generate random file names, which can be detected quickly using this method. Additionally, by reducing the dataset, you can scrutinize these items more by searching for MD5 and other signs to detect malware.

Malware has a few common places that it likes to hide, and looking at these specific places can quickly detect infection. Some examples are anything running in temporary folders. According to Niemelä (2012), some of the common locations are C:\Users\USER\Documents\, C:\system volume information\, and c:\$Recycle.Bin\ (slide 22).

Once you have used the first to techniques for detecting malware, you should have confidence that you have found at least the obvious infections on your systems. Now we can compare the "known good" configuration of these systems each day and determine what is new. This will greatly reduce the amount of data that needs analysis. It is still good to re-establish a baseline on a monthly basis to make sure nothing has slipped through.

2.3.3. Using Indicators of Compromise

When performing malware analysis on new sample you should catalog changes made to the system by the infection. Once there is a good understanding of the malware, you can generate an indicator(s) of compromise (Frazier, 2010). Indicators can be as simple as a file name or just a hash of a file. Other indicators may include file paths, process mutexes, or a service ran as a different user. To track these items and make them easily shareable, Mandiant has create an open XML framework called openIOC (http://openioc.org). They have a GUI tool that makes it easy to use the predefined

© 2013 The SANS Institute Author retains full rights.

schema for creating new entries. This allows for consistent sharing of data to different groups.

Once these items have been found, they can be queried against the database indicating other infected systems. By leveraging OHIDS, you may find systems that are lying. By definition, network IDS fails to detect anything that does not send traffic over the network.

3. Client coding designs

3.1. Client Configuration

There are several things to configure on the client before you can get started. To make these changes, you will need to edit the app.conf file in the directory where the main executable is located. The following items can be changed for your deployment: Database IP, Database Name, Database UserID, PASSWD, Registry Path to store basic information and path for the temp directory. This is covered in more detail in section 4.0

3.2. Main Function

The main function of the program has only a couple of checks and then calls the critical sub functions Analysis. Figure 1 is a map of the main function of the program.

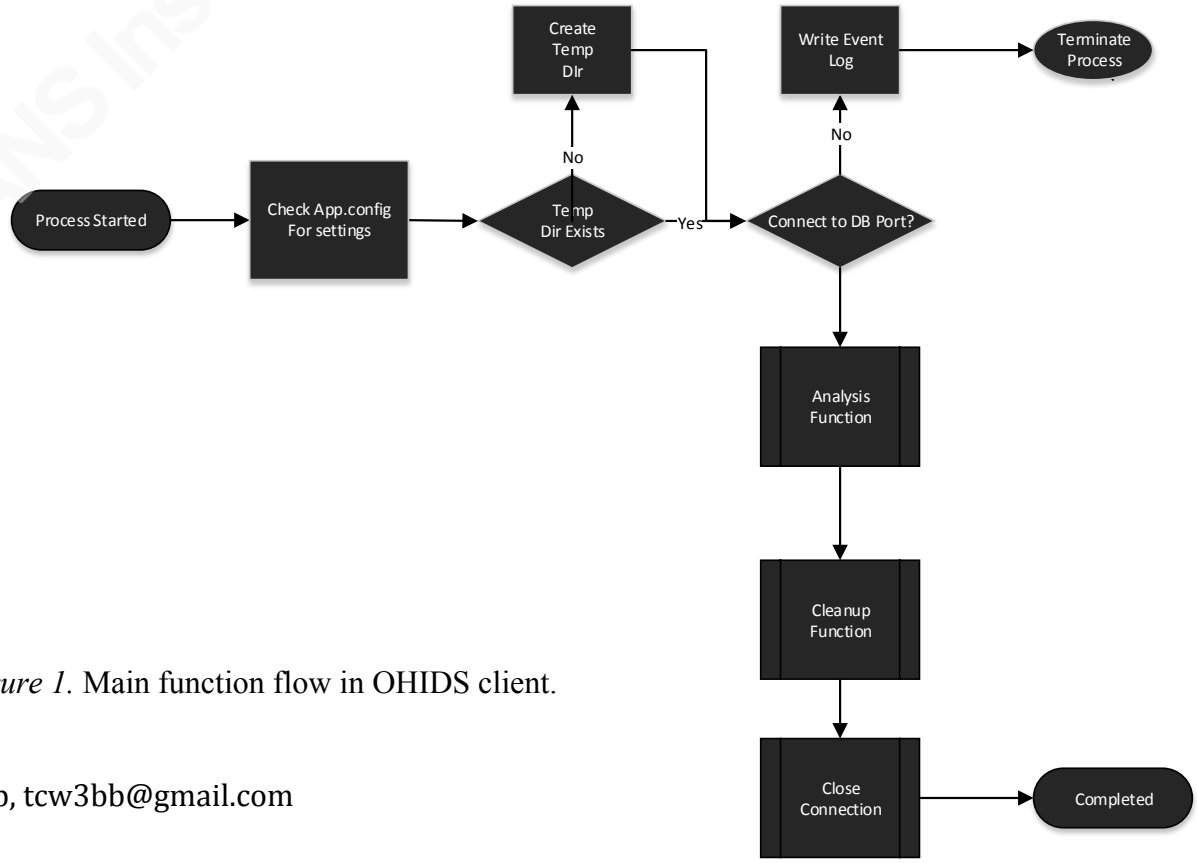


Figure 1. Main function flow in OHIDS client.

Tom Webb, tcw3bb@gmail.com

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

3.3. Input

This section will cover collecting data from the PC. The client is not allowed to read any table in the MySQL database directly (e.g. the user has limited rights to only insert) and must use stored procedures for any queries. Additionally, they can only insert into temporary tables and not in the permanent tables.

3.3.1. Analysis Function

This function collects the majority of the data for the application. The PC_Id is the unique number that is stored in the PC_Info table for each computer as the primary key. Before we can start generating results from the data, we need to know the PC_Id for this system. To do this, we call the function comid_sql that uses a stored procedure (Appendix B) to query the table and determine if an entry for the computer name already exists, if so, it will respond with the corresponding ID. If no name exists in the table, it creates a new ID and placeholder information. Once the client has determined its PC_Id, it will then proceed to collect data. Figure 2 shows how the analysis function flows.

To make sure OHIDS does not greatly effect performance, a sensitive data scan runs on a weekly basis and is set to a low priority. The last date Find_SSN.exe is run gets stored in the registry key defined in the app.config file in HKLM hive.

© 2013 The SANS Institute Author retains full rights.

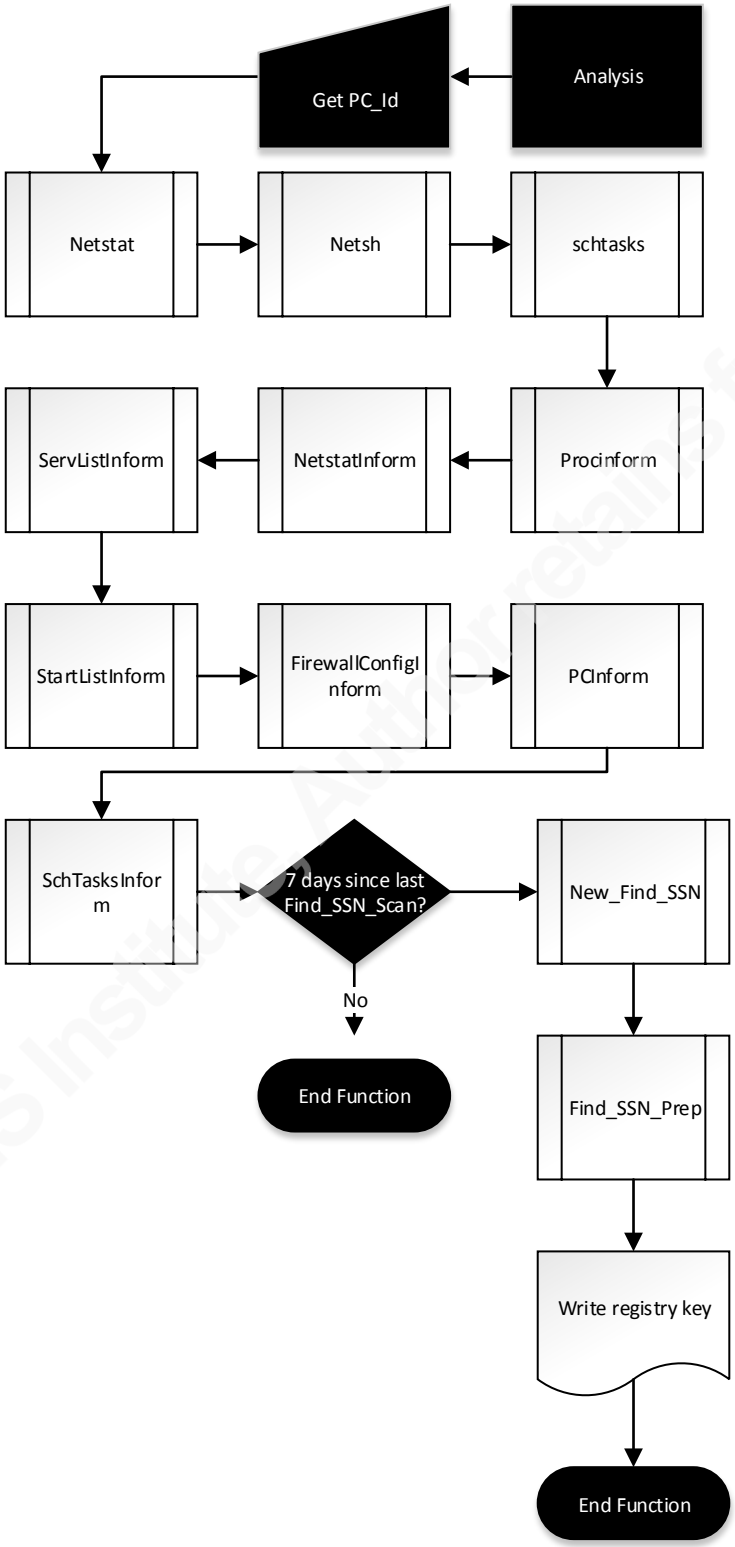


Figure 2. Analysis function flow in OHIDS client.

Tom Webb, tcw3bb@gmail.com

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

3.3.2. Netstat

To get a list of programs that have current connections and listening ports open, we use the netstat –nao command. The output redirects to a temporary file and parsed using the NetstatInform function.

3.3.3. Process list, MD5 and MAC Times

The ProcInform function collects information about the process running on the system. Additionally, it collects information on the modules loaded including MD5, PID, Parent PID and MACtimes for each process and module.

To gather a list of processes, the windows API System.Diagnostics.Process.GetProcesses() is used. WMI is used to get the parent process id (PPID) and modules for each process and results are written to a file. MAC times will be gathered for each file, using windows API, along with MD5 and File version information. Figure 3 gives a breakdown of how the function collects data.

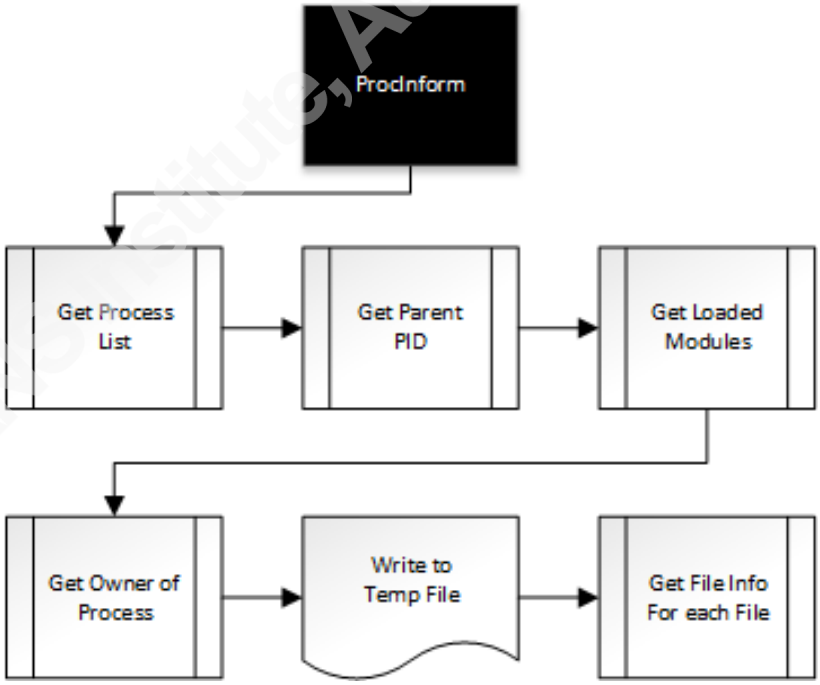


Figure 3. ProcInform function application flow in OHIDS client.

Tom Webb, tcw3bb@gmail.com

© 2013 The SANS Institute Author retains full rights.

3.3.4. Schedule Tasks

The list of scheduled tasks are generated via the built-in utility schtasks /query /v /fo csv. The output lists items any item scheduled with task scheduler or the command line at.exe. The csv file is parses and loads into the database.

3.3.5. Services

Service information is gathered using ServiceController.GetServices vb.net call. This information writes to a file, parses and loads into the database.

3.3.6. Startup List

Startup information is gathered using the WMI call Select * from Win32_StartupCommand for StartupCommand. This list includes many different items that start when the system boots or the current user logs into the system. The results are placed directly into the table using a SQL insert statement eliminating the need for a temporary file written to the host.

3.3.7. Firewall

The windows firewall rules are collected using the built-in command netsh firewall show allowed program. According to Microsoft, the policy description standard means these rules are applied when the computer is not connected to the same network as the domain. The domain description means these are applied when the system is on the same network as the domain (Microsoft, 2005). This output is redirected to a file and based on the on which OS the file is broken up into the proper format to be loaded into the database.

3.3.8. Sensitive Data Scan

OHIDS calls the sensitive data scanner find_ssn.exe two times, once for scanning the My Documents folder and the other to scan the current users' desktop, and create two separate logs. The entire user directory is not scanned due to the large number of false positives when looking at the user's temporary Internet cache and other file types stored in the folder.

© 2013 The SANS Institute Author retains full rights.

3.3.9. Error Log

Error logging is straightforward. If the system cannot connect to the database, then an event log is created in the application event log with the source of OHIDS and event ID 234. If the database can be reached, all other logs are input into the Error_Log table for troubleshooting.

3.4. Output

All functions that have output use a function with the same name and ends in sql. These functions sends the output to the database and one example of this is the netstat_sql function.

4. Database

The database should be running on MySQL 5.1 or greater on a Linux server with at least 4GB of RAM. Storage should be approximately 1GB per every 2000 PC put into the system.

4.1. Main Tables

The main table, PC_Info, is where each computer in the database must have an entry. This table contains the PC name, IP, OS version, first and last time checked in and other overall information on the system. The primary key is the PC_Id that uniquely identifies each computer that communicates to the system.

To update data in the PC_Info table, a stored procedure compares the PC_Name with the PC_Id and hashes it. This is the only table that never has any data purged from it and additional checking helps with integrity.

The Process table includes information about each process and modules loaded. The Netstat table contains results from the netstat command. Service, Startup_List, Schd_Tasks tables all contain information to detect persistence in malware. The PC_Hash table contains a list of all process and DLLs running and hashes. The Error_Log table contains errors generated by the agents when running on the systems to help with troubleshooting.

© 2013 The SANS Institute Author retains full rights.

4.2. Additional Tables

Several additional tables exist for whitelisting information; this will reduce the noise in the reports. The Good_Serv table is a list of default services installed on Windows XP and Windows 7 that was created. The Good_Hash table can be loaded with external lists of hashes or you can generate your own based on the company image. Using this table is covered in more detail in section 5.1.3. The Good_File table is used to weed out other items that you have investigated and want to remove from future reports. Items in this table must contain the full path of the executable. (e.g. insert into Good_File (Name) Value ('C:/Program Files/Trend Micro/OfficeScan Client/TmProxy.exe');) Currently, due to speed, wildcards are not supported for filtering. This makes it impossible to filter out applications installed in the user folder. To get around this, you can use the grep –v option on your output if needed.

4.3. Temp Tables

All major tables, except PC_Info, have a temporary version of the tables. This is a loading stage for comparisons, and is used to compare previous results to each other. A cron is setup to move the data daily from the temp table to the main table for storage.

4.4. Permissions needed

The SQL user for the client should have very little permissions. It should only be given insert permission to the temp tables and the ability to execute the two stored procedures used for the PC_Info table. This limits the client the ability to read any information from the tables except for what the stored procedures allow.

4.5. Stored Procedures

OHIDS has two stored procedures, get_com_id and update_comp_info. The specific code for each procedure is in Appendix B and Appendix C.

4.6. Daily Tasks

The ohids-daily-rotate script should have a link placed in the /etc/cron.daily. Before data is moved, the script purges all data in the long-term tables that is older than 45 days. The information in the PC_Info table is never removed. This script dumps the data, in CSV format, from the temp tables to the specified directory in the

© 2013 The SANS Institute Author retains full rights.

DATA_DUMP_DIR variable. This data is for historical reasons to help with incident response when a compromise is not detected for long period.

5. Deployment

5.1. Client

You may deploy the application to a folder on a network share or to a local directory. The application and dependencies will all be in one flat folder directory. Copy this directory to the place where you want it to run from. To change the default settings for the temporary file location, registry settings, and database username use the XML file in the same folder. A scheduled task should be setup via GPO and have it run daily as the System user to run the ohids.exe file.

OHIDS creates application events logs with the event id of 234 if it cannot establish a connection to the MySQL database. If you are experiencing any addition errors, you will need to check the Error_log table in the database.

5.2. Requirements and options

For most external commands run via command prompt, the results create temporary files in the specified folder on the system. The following windows utilities are used to collect data: netstat.exe, netsh.exe, schtasks.exe. Server Install

To get started, you will need a Linux system that is running MySQL. The ohids-install script is a simple script that will create the MySQL database, load the schema for the database and move scripts into the directory /usr/local/ohids. You will need to create a .my.conf in the home directory for the user you plan to run the script. You only need to include the username and password in the file for admin access to the OHIDS database. Make sure you set the permissions to 400 on this file. Edit the variables in ohids-daily-rotate.sh to make sure the script is working as intended.

© 2013 The SANS Institute Author retains full rights.

6. Mining Data

6.1. Looking for signs of compromise

As previously mentioned in the use cases, we are going to use analysis of what has changed on the computers each day. We will also use analysis of unique process running in a large environment and using known hiding places to find malware.

6.1.1. Malicious process

When detecting a malicious process, you need to know what a normal place for processes to run from is. As mentioned previously, temporary directories and system volume info is commonly used by malware. To get a list of common reports use the OHIDS report script. Make sure the Linux user has a .my.conf file in the user directory with a username and password for the database. To run the report, use ohids-reporter.sh –proc_odd. This report (Figure 4) also looks up the process hash using Team Cymru hash lookup (http://www.team-cymru.org/Services/MHR/). The Cymru results shows the percentage of Anti-virus software that detected the hash as malicious. If the results contain NO_DATA, no AV has determined the software is malicious.

```
PC_ID, Process Path, MD5, Cymru_Results
"972" "C:/WINDOWS/TEMP/DK9423.EXE" "cd6798a36930f0e224253c5db3c92d4f" NO_DATA
```

Figure 4. Sample report from Proc_Odd.

To determine unique processes running in your environment the Proc_Uniq report (Figure 5) will help you find possible random malware named processes. The report lists process in your environment that has less than five computers running the process.

```
Proc_Name  Proc_File    cat_num
ODSAgent   C:/Program Files/DVD or CD Sharing/ODSAgent.exe      3
```

Figure 5. Sample report from Proc_Uniq.

One of the most powerful ways to query this data is to compare the previous day's results to illuminate a lot of noise. The Proc_Diff report will show you only new processes running on each computer that were not running on the previous day.

© 2013 The SANS Institute Author retains full rights.

Reviewing all the various Diff reports should be part of your daily security operations process.

PC_Id	Proc_File
83	C:/Program Files/Juniper Networks/Common Files/dsNcService.exe
83	C:/Windows/system32/svchost.exe

Figure 6. Sample report from Proc_Diff.

The Proc_Date report looks for processes that are running with a file create and/or modified date of less than 48 hours. This report differs from the Proc_Diff report as malware can use the same name as another process, but the malware may not have manipulated the file dates.

PC_Id	File_Name	MD5
1187	C:/WINDOWS/ASSEMBLY/NATIVEIMAGES_V2.0.50727_64/SYSTEM.MANAGEMENT/C54FC0CAC648A174C5E35BD6589C9390/SYSTEM.MANAGEMENT.NI.DLL	0181b4c10f409299e0d8ee130ef87353

Figure 7. Sample report from Proc_Date.

If you want to dig deeper into possible malicious process with OHIDS, querying the process ID in the Process table will give you a list of all the DLLS. According to Sikorski and Honig (2012), you can use the loaded DLLs to determine some of the capabilities of the malware (p. 17).

6.1.2. Detecting Persistence

The Start_List table, Sch_Tasks and Service_List are the key tables when looking for items where the malware will reload when the system restarts. The Start_Loc report, queries the Start_List table, is the lowest volume and tends to have the highest accuracy for detecting infections. Like the process version of this report, the Start_Diff looks for typical places where malware likes to hide but only shows the difference between the same computers for the past 48 hours.

PC_Id	Command
15	Facebook Messenger.lnk
15	OneNote 2007 Screen Clipper and Launcher.lnk

Figure 8. Sample report from Start_Loc.

© 2013 The SANS Institute Author retains full rights.

PC_Id	Cname	Command
1252	1D5N1	C:/PROGRA~1/SAFECO~1/scClient.exe

Figure 9. Sample report from Start_Diff.

6.1.3. Compare Software Hashes

Looking for software that is running the same reported version, but a different hash, can be tricky. You are relying on the developers to update the file version metadata any time a change is made. The Hash_Comp report (Figure 10) compares MD5 hashes of running exes and DLL's with the same version number. If a hash does not match, it will display the name. I use this report as a final effort when other analysis is not working out. The success rate for detection has been very low on this report in my uses.

Filename	Version	MD5
ADL.FOUNDATION.DLL	2.0.3299.28586	0a8e6b6caac2d01de4c56b15022a6b3b
ADL.FOUNDATION.DLL	2.0.3299.28586	1bbdbd33cde07ba454923247a9b0be12
ADL.FOUNDATION.DLL	2.0.3299.28586	1d5a364193eed5a97803b95377ac15ee

Figure 10. Sample report from Hash_Comp.

The most well known hash dataset is the National Institute of Standards and Technology's National Software Reference Library(2009). You can also use their Knoppix CD to create your own list from you standard desktop image. By loading this information into the Good_Hash table, you can use it to reduce the noise in these reports. This can be done by comparing the results of the MD5s to the items listed in the Good_Hash table and only displaying items that do not match. If you want to have any your queries filter based on this table, you can add "WHERE MD5 not in (select DISTINCT MD5 from Good_Hash)" to the end of the query. This is already done in the Hash_Comp report.

If you are allowed to send data offsite or if you want to uses these hashes with other tools, the nsrllookup (nsrllookup, 2013) software allows you to send hashes to a database for lookups. The project is hosting a copy of the NSRL hashes to query for free, but if you use a third party, make sure you verify these results. By dumping the results of

© 2013 The SANS Institute Author retains full rights.

the Hash_Comp report into a file and then running the following command "nsrllookup -s nsrl.kyr.us <hash" you get a list of only unknown hashes from the database.

6.2. Personally Identifiable Information Scan results

As mentioned previously in the use case 1.6.1, finding out which system has sensitive data is critical to determine what level of response you should provide to a compromised system. To get a list of sensitive data for a specific computer, use the –Ssn_Comp switch followed by the computer name (e.g. –Ssn_Comp 'PC_ID'). The total includes the total number of SSNs and/or credit card numbers found on the system. The SSN_Top report include the top 25 highs totals.

Date	PC_Id	Cname	Total	IP
2013-03-01	586	KDJFLWGQ1	12644	1.2.3.4
2013-03-08	271	KLDJFLOEIU	10730	1.2.3.5

Figure 11. Sample report from SSN_Top.

This report will give you a list of files that contain possible social security numbers and/or credit card number count on the system. We do not keep track of the actual SSN/CCN found on the system. An additional indicator, if these are false positives, are location of the files and file names. For example, if you have a file that shows 5,000 PII count and it's called mathematical_analysis_of_water_sample.xls chances are this is a false positive. The spreadsheet likely has many numbers in it for data samples. However, if the file name is Q3_finance_dept.xls it is likely to be a true positive and escalate this incident appropriately. You can also call the individual armed with this information and ask specific questions about these files. In this case, you will likely get an accurate response from the user.

7. Proactive Steps to prevent breaches

7.1. Remove Unnecessary Sensitive data

Incidents will continue to happen in every environment and the best protection from data loss is to remove unnecessary data completely from system. A very effective way to remediating this issue is generating a top 25-user report on a weekly basis (e.g. –SSN_TOP). With this data, you can create a helpdesk ticket and have them contact the

© 2013 The SANS Institute Author retains full rights.

user about the data in the specific report. In addition, you can have this report sent to management so they understand the higher risk areas. In many cases, the user had forgotten the data was there or did not need to keep it on their system. If they need to keep it, you should work to determine acceptable recommendation to mitigate risks

7.2. Audit for unauthorized software

The Proc_Date report gives you a list of new software installed in the last 48 hours that was running on the system. This report can detect malware, but also detecting programs that are not allowed by policy (e.g., P2P applications, Cloud file sharing). If you have users running portable apps you will be able to detect this by looking at the path where the executable is running from.

8. Future plans

8.1. Gather Specific Event logs

Anyone who has looked at Windows event logs know they are very noisy. If you do not already have a centralized logging, then gathering a small number of events can be useful. Placing the event IDs you want to capture into the xml config will allow you to specify what logs you want to get back into the system. The NSA white paper "Spotting the Adversary with Windows Event Log Monitoring" contains a comprehensive listing of what event logs should be monitored (NSA, 2013). For Windows 7 systems, the following event ID's will initially be the items of interest to collect: 4740, 4624, 4625, 865, 8003, 8004, 8006, and 8007.

8.2. On demand Scanning/Service

Having OHIDS run as a service would allow it to check-in at a short interval to the database and determine if it should run a new scan. This would then allow me to build a simple API to be used by an IDS to request data to be gathered on the host immediately. The data could then be quickly reviewed to determine if a network attack was successful.

8.3. API Hooking

Software like El Jefe (El Jafe, 2013), which hooks the create process API and records the information to a MySQL server, has been around for a couple of years. By

© 2013 The SANS Institute Author retains full rights.

logging these specific calls, incident responders will know what processes are ran on a system and what started the process. Adding a similar functionality to this code would give CIRTs near real-time data on what is running in the environment. This is a much longer goal, as the on demand scanning will be easier to implement.

9. Conclusion

Network IDS will continue to be used for a long time, while integrating NIDS information into security information and event management(SIEM) systems has made correlation of successful attacks easier, but collecting key information from the hosts is still the most beneficial information for CIRTs. Using OHIDS in conjunction with NIDS makes it possible to detect an incident via the NIDS then gather detailed analysis and determine the amount of sensitive data stored on the system. With the hash of the malware, other online resources may have already performed an analysis of the executable. If so, you can quickly determine if the capabilities of the malware allows access to any sensitive data found on the system. All this can be completed before you start your forensics collection of evidence. With the provided data, you can determine if it is necessary to do any additional investigation or begin the clean up process. By eliminating the need for in-depth analysis in some incidents or speeding up the forensics process, OHIDS enables CIRTs to handle incidents more efficiently and enable them to spend resources on the more complicated cases.

Tom Webb, tcw3bb@gmail.com

© 2013 The SANS Institute Author retains full rights.

10. References

Bejtlich, R. (2006). *Extrusion Detection*. Upper Saddle River, NJ: Pearson Education, Inc.

Caswell, B., Beale, J., & Baker, A. R. (2007). *Snort®: IDS and IPS Toolkit.* Burlington, MA: Syngress Publishing, Inc.

Dyrmose, M. (2013, January 5). *Beating the IPS*. Retrieved from http://www.sans.org/reading_room/whitepapers/intrusion/beating-ips_34137

El Jefe (2013). Retrieved from http://www.immunityinc.com/products-eljefe.shtml

Frazier, M. (2010, January 26). *Combat the APT by Sharing Indicators of Compromise.* Retrieved from https://www.mandiant.com/blog/combat-apt-sharing-indicators-compromise

Find_SSNs (2009). Retrieved from http://www.security.vt.edu/resources_and_information/find_ssns.html

Help: Understanding Windows Firewall Profiles. (2005, January 25). Retrieved from http://technet.microsoft.com/en-us/library/cc739685(v=ws.10).aspx

Innella, P. (2001, November 16). *The Evolution of Intrusion Detection Systems*. Retrieved from http://www.symantec.com/connect/articles/evolution-intrusion-detection-systems

National Security Agency, Central Security Services. (2013). *Spotting the Adversary with Windows Event Log Monitoring* (TSA-13-1004-SG). Retrieved from website http://www.nsa.gov/ia/_files/app/Spotting_the_Adversary_with_Windows_Event_Log_Monitoring.pdf

Niemelä, J. (2012). *Making Life Difficult for Malware*. Retrieved from http://www.blackhat.com/docs/webcast/bh-wb-May12-Making_Life_Difficult_for_Malware.pdf

Nsrllookup (2013). Retrieved from http://rjhansen.github.io/nsrllookup/

Sikorski, M., & Honig, A. (2012). *Practical Malware Analysis*. San Francisco, CA: No Starch Press, Inc.

Smaha, S. (1996). *Computer Misuse and Anomaly Detection – IV*. Retrieved from http://seclab.cs.ucdavis.edu/projects/cmad/4-1996/pdfs/Smaha.pdf

U.S. Department of Commerce, National Institute of Standard and Technology. (2009). *National Software Reference Library*. Retrieved from http://www.nsrl.nist.gov/Downloads.htm#isos

Tom Webb, tcw3bb@gmail.com

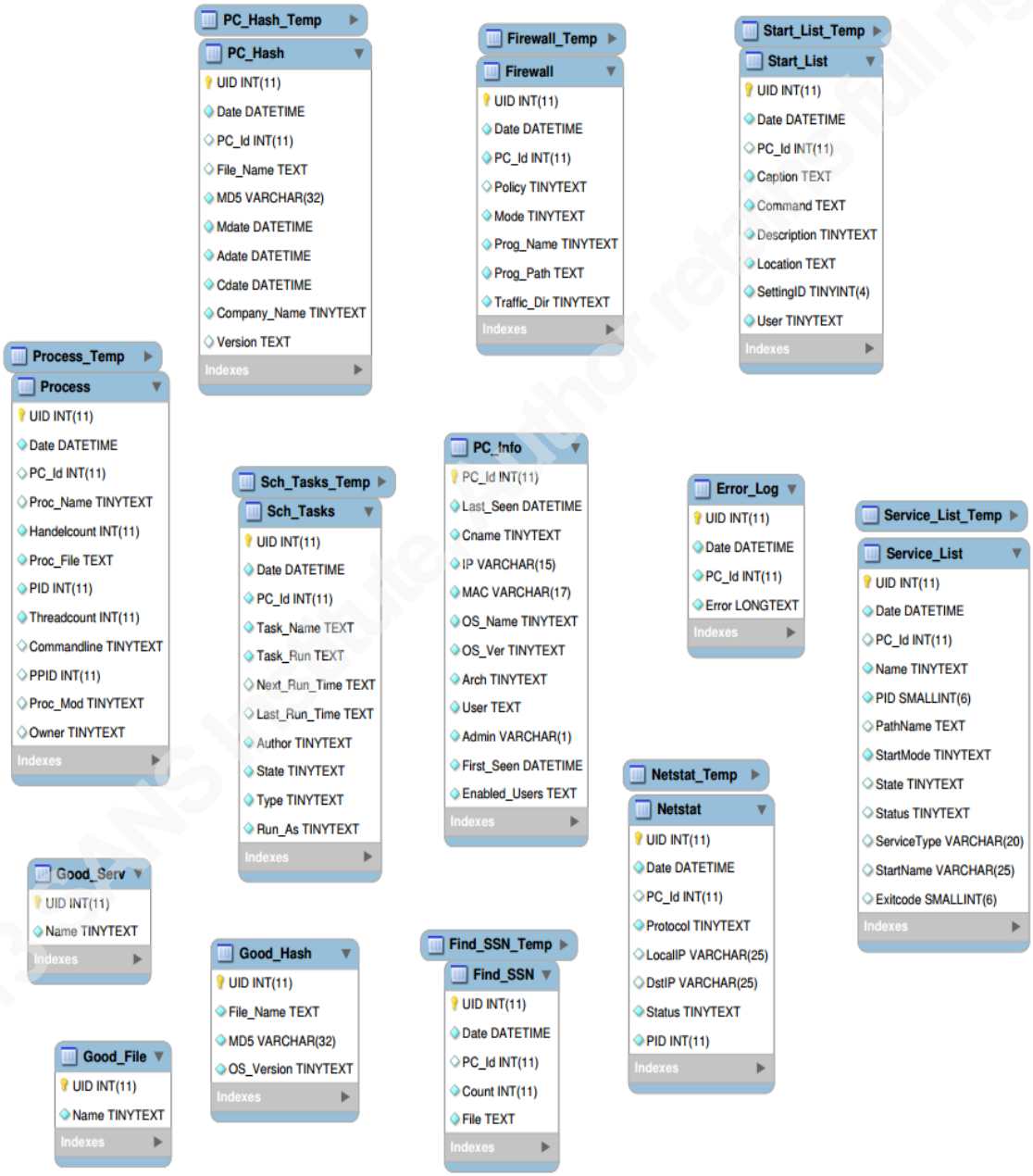
© 2013 The SANS Institute Author retains full rights.

© 2013 SANS Institute, Author retains full rights.

Tom Webb, tcw3bb@gmail.com

© 2013 The SANS Institute Author retains full rights.

Appendix A Database table layout



Tom Webb, tcw3bb@gmail.com

© 2013 The SANS Institute Author retains full rights.

Appendix B get_com_id

'Latest version of the code is available at https://code.google.com/p/open-source-host-based-ids/

```
DROP PROCEDURE IF EXISTS `OHIDS`.`get_com_id`;
DELIMITER $$
CREATE PROCEDURE `OHIDS`.`get_com_id`(IN input VARCHAR(30), OUT
compidnum INT)
READS SQL DATA
BEGIN
DECLARE curdate_val DATETIME;
DECLARE compidnum int;
SET @input=input;
SET @compidnum= NULL;

prepare compid from
  'select PC_Id INTO @compidnum from PC_Info where Cname= ? limit 1' ;

execute compid USING @input;


IF @compidnum IS NULL THEN
      SELECT NOW() INTO curdate_val;
      INSERT INTO ITSO_PC_IR.PC_Info
(PC_Id,Last_Seen,Cname,IP,MAC,OS_Name,OS_Ver,Arch,User,Admin,First_Seen,Ena
bled_Users)
VALUES('0',curdate_val,INPUT,'0.0.0.0','::','Windows','unknown','unknown','unknown','
unknown',curdate_val,'me');
END IF;

execute compid USING @input;

select @compidnum;

DEALLOCATE PREPARE compid;
```

Tom Webb, tcw3bb@gmail.com

© 2013 SANS Institute, Author retains full rights.

Appendix C Update_comp_info

'Latest version of the code is available at https://code.google.com/p/open-source-host-based-ids/

```
DROP PROCEDURE IF EXISTS `OHIDS`.`update_comp_info`;
DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `update_comp_info`(IN
authcode CHAR(32), IN PCid text,  IN PCdate datetime, IN IP varchar(15), IN MAC
varchar(17),
IN OS_NAME tinytext, IN OS_VER tinytext, IN Arch tinytext, IN PCUser text, IN
Admin varchar(1), IN Enabled_Users text )
   READS SQL DATA
BEGIN
prepare getcompname from
'select Cname INTO @mysqlcompname from PC_Info where PC_Id= ? limit 1' ;

SET @PCid=PCid;
execute getcompname USING @PCid;

select md5(CONCAT (@mysqlcompname , @PCid)) into @sql_auth_code;

SET @PCdate=PCdate;
SET @IP=IP;
SET @MAC=MAC;
SET @OS_Name=OS_Name;
SET @OS_VER=OS_VER;
SET @Arch=Arch;
SET @PCUser=PCUser;
SET @Admin=Admin;
SET @Enabled_Users=Enabled_Users;
SET @PCid=PCid;

IF authcode LIKE @sql_auth_code THEN
      prepare compupdate from
'UPDATE PC_Info SET Last_Seen= ?, IP= ?,  MAC=?, OS_Name= ?,  OS_Ver= ?,
Arch= ?, User= ?, Admin= ?, Enabled_Users= ? WHERE PC_ID= ?';
execute compupdate USING @PCdate, @IP, @MAC, @OS_Name, @OS_VER,
@Arch, @PCUser, @Admin, @Enabled_Users, @PCid ;

DEALLOCATE PREPARE compupdate;
else
select "No Match";
```

Tom Webb, tcw3bb@gmail.com

© 2013 The SANS Institute Author retains full rights.

```
END IF;
DEALLOCATE PREPARE getcompname;
END
```

Tom Webb, tcw3bb@gmail.com

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

Appendix D OHIDS Client Code

```
'*****************************************************************
‘OHIDS 1.0
'Authors - Tom Webb (tcw3bb@gmail.com),
‘Previous Contributer Vipul Gupta (vipulgupta0@gmail.com)
'This script collects system specific data for forensic analysis
'This script will work on vb.net 4.0.
‘https://code.google.com/p/open-source-host-based-ids/
'*****************************************************************
'*****************************************************************
'Importing required modules
'*****************************************************************
Imports System.Net.NetworkInformation
Imports System.Management
Imports System.Security.Principal
Imports System.Text.RegularExpressions
Imports System.Threading
Imports System.IO
Imports System.ServiceProcess
Imports Microsoft.Win32
Imports MySql.Data.MySqlClient
Imports System.Diagnostics

Module Module1
   '*****************************************************************
   'Declare constants
   '*****************************************************************
   Dim Machine As String = System.Environment.MachineName
   'Dim mydate = DateTime.Now.ToString("yyyy-MM-dd") 'new
   Dim mydate = Format(System.DateTime.Now, "yyyy-MM-dd HH:mm:ss")
   Const ForReading = 1
   Const ForAppending = 8
   Const ForOverWriting = 2
   Const OpenAsASCII = 0
   Const OpenAsUnicode = -1
   Const OpenUsingDefault = -2
   Const OverWriteExisting = True
   Const HKLM = &H80000002 'HKEY_LOCAL_MACHINE

   Dim nowdate As Date = System.DateTime.Now.Date
   Dim RunCmds As Boolean
   Dim wshNetwork As Object = CreateObject("WScript.Network")
   Dim strComputerName As String = Machine
   Dim oShell As Object = CreateObject("WScript.Shell")
```


© 2013 The SANS Institute Author retains full rights.

```
    Dim dir = AppDomain.CurrentDomain.BaseDirectory


    Dim strOutPutStream
    Dim strDate As Date
    Dim LocalUserName As String
    Dim osArch As String = ""
    Dim PC_ID As Int32
    Dim Registery_Path As String =
System.Configuration.ConfigurationManager.AppSettings("Registery_Path")
    Dim Temp_Path As String =
System.Configuration.ConfigurationManager.AppSettings("Temp_Path")
    '*************************************************************
    'MYSQL Variables
    '*************************************************************
    Dim server As String =
System.Configuration.ConfigurationManager.AppSettings("DB_SERVER")
    Dim database As String =
System.Configuration.ConfigurationManager.AppSettings("DB_Database_Name")
    Dim userid As String =
System.Configuration.ConfigurationManager.AppSettings("DB_UserId")
    Dim password As String =
System.Configuration.ConfigurationManager.AppSettings("DB_PASSWD")
    Dim ConnectionString = "server=" & server & ";" & "user id=" & userid & ";" &
"password=" & password & ";" & "database=" & database & ";" & "SSL
Mode=Required" & ";" & "check parameters=false"

    Dim MysqlConn As MySqlConnection


    '*****************************************************************
    'Main Module, entry to the program
    '*****************************************************************
    Sub Main()

        If (Not System.IO.Directory.Exists(Temp_Path)) Then
            System.IO.Directory.CreateDirectory(Temp_Path)
        End If

        If QueryPort() = True Then
            Analyze()
            'After data collection, if no errors are encountered (the flag
            'is True), then write to registry
            Cleanup()            'Cleanup, Rename Files and Copy to destination
            MySQLCloseConnection() ' last thing to run
```


© 2013 The SANS Institute Author retains full rights.

```
        Else
            'Bow out because server is not open."
        End If

    End Sub

    '*****************************************************************
    'The main data collection and analysis module, makes necessary
    'function calls and starts data collection tools
    '*****************************************************************
    Function Analyze()
        MySQLOpenConnection(ConnectionString) 'open sql connection
        comid_sql(Machine) ' Get PC_ID from the data to use in output

        'Starting netstat, netsh, and schtasks
        Process.Start("cmd", "/c netstat -nao >" & Temp_Path & "\netstat.txt")
        Process.Start("cmd", "/c netsh firewall show allowedprogram >" & Temp_Path &
"\firewall.txt")
        Process.Start("cmd", "/c schtasks /query /v /fo csv > " & Temp_Path & "\s.csv")

        ProcInform()            'Get the Running Processes Related Information
        NetstatInform()         'Get the Netstat Information
        ServListInform()        'Get the Service List Information
        StartListInform()       'Get the Startup List Information
        FirewallConfigInform()  'Get the Windows Firewall Configuration Information
        PCInform()              'Get the PC Information
        SchTasksInform()        'Get the Schedule Tasks Information


        Dim oReg As Object = GetObject("winmgmts://" & strComputerName &
"/root/default:StdRegProv")
        Dim strError = oReg.CreateKey(HKLM, Registery_Path) 'Create Key

        If ReadRegistry(HKLM, Registery_Path, "LastFindSSN", strDate) IsNot "False"
Then
            If DateDiff("d", ReadRegistry(HKLM, Registery_Path, "LastFindSSN", strDate),
nowdate) >= 7 Then
                New_FindSSNs()
                find_ssn_prep()        'Prep Files for upload
                WriteRegistry("HKLM\" & Registery_Path & "LastFindSSN", nowdate,
"REG_SZ")

            Else
                'Bow out.
            End If
        End If
```


© 2013 The SANS Institute Author retains full rights.

```
        Return Err()
    End Function

    Function New_FindSSNs()


        If (My.Computer.Info.OSFullName.Contains("Windows XP")) Then
            Dim WinXPFindSSN As New System.Diagnostics.Process
            Dim ssnargs As String = " -p " & """c:\Documents and Settings\" &
LocalUserName & "\My Documents""" & " -o " & Temp_Path & " -t csv -a"
            Try

                Dim SI As New ProcessStartInfo(dir & "Find_SSNs.exe", ssnargs)

                WinXPFindSSN.StartInfo = SI
                WinXPFindSSN.Start()
                WinXPFindSSN.WaitForExit()

                'Rename the file before its over written for the second scan of desktop

                Dim OldName, NewName As String
                OldName = Temp_Path & "\Find_SSNs.csv"
                NewName = Temp_Path & "Find_SSNs1.csv" ' Define file names.
                Rename(OldName, NewName)   ' Rename file.
                '   System.IO.File.Delete("c:\temp\Find_SSN.txt")

            Catch ex As Exception
                error_sql("Error in findssn function file rename to Find_SSN1.csv:" &
ex.Message)
            End Try

            Dim WinXPFindSSNDekstop As New System.Diagnostics.Process
            Dim ssnargsdesk As String = " -p " & """c:\Documents and Settings\" &
LocalUserName & "\Desktop""" & " -o " & Temp_Path & " -t csv -a"
            Dim SI2 As New ProcessStartInfo(dir & "Find_SSNs.exe", ssnargsdesk)

            WinXPFindSSNDekstop.StartInfo = SI2
            WinXPFindSSNDekstop.Start()
            WinXPFindSSNDekstop.WaitForExit()

        Else ' Windows Vista or greater

            Dim Win7FindSSN32 As New System.Diagnostics.Process
            Dim ssnargs32 As String = " -p " & """c:\Users\" & LocalUserName & "\Local
Documents""" & " -o " & Temp_Path & " -t csv -a"
```


© 2013 The SANS Institute Author retains full rights.

```
        Try
            Dim SI As New ProcessStartInfo(dir & "Find_SSNs.exe", ssnargs32)

            ' Console.WriteLine(dir & "Find_SSNs.exe" & ssnargs32)
            ' Console.ReadLine()

            Win7FindSSN32.StartInfo = SI
            Win7FindSSN32.Start()
            Win7FindSSN32.WaitForExit()

            'Rename the file before its over written

            Dim OldName, NewName As String
            OldName = Temp_Path & "\Find_SSNs.csv"
            NewName = Temp_Path & "\Find_SSNs1.csv" ' Define file names.
            Rename(OldName, NewName)   ' Rename file.

        Catch ex As Exception
            error_sql("Error in findssn function file rename to Find_SSN1.csv:" &
ex.Message)
        End Try

        Dim Win7FindSSN32desk As New System.Diagnostics.Process
        Dim ssnargs32desk As String = " -p " & """c:\Users\" & LocalUserName &
"\Desktop""" & " -o " & Temp_Path & " -t csv -a"
        Try
            Dim SI2 As New ProcessStartInfo(dir & "Find_SSNs.exe", ssnargs32desk)


            Win7FindSSN32.StartInfo = SI2
            Win7FindSSN32.Start()
            Win7FindSSN32.WaitForExit()
        Catch ex As Exception
            error_sql("Error in findssn function file rename to Find_SSN1.csv:" &
ex.Message)
        End Try

    End If

    Return Err()

End Function
'*****************************************************************
'FIND_SSN_PREP
'Gets the results from ssn find ready for SQL insertion
'*****************************************************************
```


© 2013 The SANS Institute Author retains full rights.

```
Function find_ssn_prep()
    Try
        If File.Exists(Temp_Path & "\Find_SSNs1.csv") Then
            text_combine(Temp_Path & "\Find_SSNs.csv", Temp_Path &
"\Find_SSNs1.csv") 'Combine the two results file into one find_ssn1.csv
        Else
            Dim oldname = Temp_Path & "\Find_SSNs.csv"
            Dim newname = Temp_Path & "\Find_SSNs1.csv"
            Rename(oldname, newname) ' Always have a file named Find_SSNs1.csv it
results from the other scan
        End If

        Dim ssnline As StreamReader
        Dim myline As String

        ssnline = New StreamReader(Temp_Path & "\Find_SSNs1.csv", FileMode.Open)

        Do While ssnline.Peek >= 0
            myline = ssnline.ReadLine() ' set myline as varible for each line read
            ' Console.WriteLine(myline)
            '  Console.Read()
            If Not myline.Contains("#") Then ' remove file header of #
                If Not myline.Contains("NO") Then ' Removes lines where no data found
                    Dim Values() As String = Split(myline, ",")
                    '  Console.WriteLine(Values(0) + Values(2)) ' we only need the 1st and
3rd values
                    findssn_sql(mydate, Values(0), Values(2))
                End If
            End If
        Loop
        ssnline.Close()
        ssnline.Dispose()
    Catch ex As Exception
        error_sql("Error in findssn_prep function:" & ex.Message)
    End Try
    Return vbNullString 'stops vb.net error
End Function
'*******************************************************************
'Cleanup, Rename Files and Copy to destination
'*******************************************************************
Function Cleanup()

    Try
        System.IO.File.Delete(Temp_Path & "\s.csv")
        System.IO.File.Delete(Temp_Path & "\s_distinct.csv")
        System.IO.File.Delete(Temp_Path & "\firewall.txt")
```


© 2013 The SANS Institute Author retains full rights.

```
        System.IO.File.Delete(Temp_Path & "\ua.csv")
        System.IO.File.Delete(Temp_Path & "\schTaskFinal.csv")
        System.IO.File.Delete(Temp_Path & "\Find_SSNs")
        System.IO.File.Delete(Temp_Path & "md5sum.csv")
        System.IO.File.Delete(Temp_Path & "\process.csv")
        System.IO.File.Delete(Temp_Path & "\Find_SSNs.csv")
        System.IO.File.Delete(Temp_Path & "\Find_SSNs1.csv")
    Catch ex As Exception

    End Try

    Dim sdatafile2 As String
    sdatafile2 = Temp_Path + Machine + ".File_Content.csv"
    'Delete any older file with the same name
    If System.IO.File.Exists(sdatafile2) Then
        System.IO.File.Delete(sdatafile2)
    End If

    'If errors are encountered during data collection, set flag to False so that registry
    'is not written, else set it to True
    If Err.Number <> 0 Then
        RunCmds = False
    Else
        RunCmds = True
    End If
    Return RunCmds
End Function

'*****************************************************************
'Get the Schedule Tasks Information
'*****************************************************************
Function SchTasksInform()
    Thread.Sleep(8000)
    Dim slines As String() = IO.File.ReadAllLines(Temp_Path & "\s.csv")
    Dim dlines As String() = slines.Distinct.ToArray()
    'Get unique Scheduled tasks and write to a temp file
    IO.File.WriteAllLines(Temp_Path & "\s_distinct.csv", dlines)
    Dim schTaskFinal As String = Temp_Path & "\schTaskFinal.csv"
    Dim schTaskFinalWriter As New System.IO.StreamWriter(schTaskFinal)
    'From the temp file, write in proper format to be written to the final file
    Dim tasklines As String() = IO.File.ReadAllLines(Temp_Path & "\s_distinct.csv")
    Dim schcount As Integer = 0
    Dim schline As String()

    'Scheduled Tasks are differently arranged in Windows 7 and Windows XP
    'Following code does the formatting based on the OS
```


© 2013 The SANS Institute Author retains full rights.

```
        'For Windows 7
        If (My.Computer.Info.OSFullName.Contains("Windows 7")) Then
            For schcount = 1 To tasklines.Count - 1
                'The scheduled task fields are surrounded by double quotes, split them
                'based on the double quotes
                schline = Regex.Split(tasklines(schcount), """")
                If schline(19).StartsWith(" ") Then
                    schtask_sql(mydate, PC_ID, schline(3), schline(17), schline(5), schline(11),
schline(15), schline(25), schline(39), schline(31))
                Else
                    schtask_sql(mydate, PC_ID, schline(3), schline(17), schline(5), schline(11),
schline(15), schline(23), schline(37), schline(29))
                End If
            Next
            'For Windows XP
        ElseIf My.Computer.Info.OSFullName.Contains("Windows XP") Then
            For schcount = 2 To tasklines.Count - 1
                schline = Regex.Split(tasklines(schcount), """")
                Dim ldate As String
                Dim ndate As String
                Try
                    'If the dates are in a different format, convert them to maintain consistency
                    Dim mdate As Date = CDate(schline(9))
                    Dim ntime As Date = CDate(schline(5))
                    ndate = ntime.ToString
                    ldate = mdate.ToString
                Catch ex As Exception
                    ndate = schline(9).ToString
                    ldate = schline(5).ToString
                End Try
                schtask_sql(mydate, PC_ID, schline(3), schline(17), ndate, ldate, schline(13),
schline(23), schline(25), schline(37))
            Next
        End If
        schTaskFinalWriter.Close()

        Return Err()
    End Function
    '*******************************************************************
    'Get the Windows Firewall Configuration Information
    '*******************************************************************
    Function FirewallConfigInform()
        'Windows Firewall Configuration Information
        Dim Machine As String = System.Environment.MachineName
        Thread.Sleep(8000)
```


© 2013 The SANS Institute Author retains full rights.

```
Dim sfresult As String()
Dim dors As String = ""
Dim bline() As String
Dim sr As String()
Dim sep() As String

'Formatting is different in Windows 7 and XP
'Windows XP does not return the Traffic Direction Information

If (My.Computer.Info.OSFullName.Contains("Windows 7")) Then
   Try
      Dim rflines As String() = IO.File.ReadAllLines(Temp_Path & "\firewall.txt")
      For i = 0 To rflines.Length - 1
         sfresult = Split(rflines(i), " ")
         If sfresult(0).ToString = "Allowed" Then
            i = i + 3
            dors = sfresult(4)
            sr = Split(rflines(i), " ")
            If sr(0).ToString = "Enable" Or sr(0).ToString = "Disable" Then
               bline = Regex.Split(rflines(i), "\s\s+")
               sep = Regex.Split(bline(2), " / ")
               'This writes the first line that matches criteria
               firewall_sql(mydate, PC_ID, dors, bline(0), sep(0), sep(1), bline(1))
            End If
         End If
         If sfresult(0).ToString = "Enable" Or sfresult(0).ToString = "Disable" Then

            bline = Regex.Split(rflines(i), "\s\s+")
            If bline(1).ToString = "Inbound" Or bline(1).ToString = "Outbound" Then
               sep = Regex.Split(bline(2), " / ")
               'This writes the subsequent lines that match the criteria
               firewall_sql(mydate, PC_ID, dors, bline(0), sep(0), sep(1), bline(1))
            End If
         End If
      Next
   Catch ex As Exception
      error_sql("Error in FirewallconfigInform function:" & ex.Message)
   End Try

ElseIf My.Computer.Info.OSFullName.Contains("Windows XP") Then
   Try
      Dim rflines As String() = IO.File.ReadAllLines(Temp_Path & "\firewall.txt")
      For i = 0 To rflines.Length - 1
         sfresult = Split(rflines(i), " ")
         If sfresult(0).ToString = "Allowed" Then
            i = i + 3
```


© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

```
                dors = sfresult(4)
                sr = Split(rflines(i), " ")
                If sr(0).ToString = "Enable" Or sr(0).ToString = "Disable" Then
                    bline = Regex.Split(rflines(i), "\s\s+")
                    sep = Regex.Split(bline(1), " / ")
                    'This writes the first line that matches the criteria
                    firewall_sql(mydate, PC_ID, dors, bline(0), sep(0), sep(1), bline(1))
                End If
            End If
            If sfresult(0).ToString = "Enable" Or sfresult(0).ToString = "Disable" Then
                bline = Regex.Split(rflines(i), "\s\s+")
                sep = Regex.Split(bline(1), " / ")
                'This writes the subsequent lines that match the criteria
                firewall_sql(mydate, PC_ID, dors, bline(0), sep(0), sep(1), bline(1))
            End If
        Next
    Catch ex As Exception
        error_sql("Error in FirewallconfigInform function:" & ex.Message)
    End Try
End If

Return Err()
End Function
'*****************************************************************
'Get the Startup List Information
'*****************************************************************
Function StartListInform()
    Dim obj_WMI, objStartup, rQuery
    'Creating a WMI object
    'impersonate means that the current user's permissions will be used by WMI
    'We will be running the script as a whole with elevated permissions, so that is what
    'matters in the end, this is why defaults used here
    'In a nutshell, connecting to WMI using defaults
    obj_WMI =
GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2")
    rQuery = obj_WMI.ExecQuery("Select * from Win32_StartupCommand")

    For Each objStartup In rQuery
        start_list_sql(mydate, PC_ID, objStartup.Caption(), objStartup.Command(),
objStartup.Description(), objStartup.Location(), objStartup.SettingID(),
objStartup.User())
        'Console.Write(start_list_sql)
        'Console.Read()
    Next
    ' startListCsvWriter.Close()
```


© 2013 The SANS Institute Author retains full rights.

```
        Return Err()
    End Function
    '*****************************************************************
    'Get the Service List Information
    '*****************************************************************
    Function ServListInform()


        Dim scServices() As ServiceController
        scServices = ServiceController.GetServices()
        Dim scTemp As ServiceController
        For Each scTemp In scServices
            Dim wmiService As ManagementObject
            wmiService = New ManagementObject("Win32_Service.Name='" +
scTemp.ServiceName + "'")
            wmiService.Get()
            service_list_sql(mydate, wmiService("Name").ToString,
wmiService("ProcessID").ToString, wmiService("PathName").ToString,
wmiService("StartMode").ToString, wmiService("State").ToString,
wmiService("Status").ToString, wmiService("ServiceType").ToString,
wmiService("StartName").ToString, wmiService("ExitCode").ToString)

        Next scTemp
        Return Err()


    End Function
    '*****************************************************************
    'Get the Netstat Information
    '*****************************************************************
    Function NetstatInform()


        Thread.Sleep(5000)
        Try
            Dim rlines As String() = IO.File.ReadAllLines(Temp_Path & "\netstat.txt")
            Dim sResult As String()
            'Formatting as needed
            For Each rline In rlines
                rline = Trim(rline)
                sResult = Split(rline, " ")
                If (sResult(0).ToString = "TCP") Then
                    Dim tcpr As String = rline
                    Dim tarr As String() = SplitFields(tcpr)
                    netstat_sql(mydate, PC_ID, tarr(0), tarr(1), tarr(2), tarr(3), tarr(4))
```


© 2013 The SANS Institute Author retains full rights.

```
            ElseIf (sResult(0).ToString = "UDP") Then
                Dim udpr As String = rline
                Dim uarr As String() = SplitFields(udpr)
                netstat_sql(mydate, PC_ID, uarr(0), uarr(1), uarr(2), "", uarr(3))

            End If
        Next
        System.IO.File.Delete(Temp_Path & "\netstat.txt")
    Catch ex As Exception
        error_sql("Error in NetstatInform function:" & ex.Message)
    End Try
    Return Err()

End Function
'*****************************************************************************
'Get the Running Processes Related Information
'*****************************************************************************
Function ProcInform()
    Dim oProcesses() As Process = System.Diagnostics.Process.GetProcesses()
    Dim pro As New Process()
    Dim csvFile As String = Temp_Path & "\process.csv"
    Dim md5values As String = Temp_Path & "\md5sum.csv"
    Dim md5writer As New System.IO.StreamWriter(md5values)
    'A temp file that will contain module names for all processes
    'later we will select the uniques out of it
    Dim tempHashFile As String = Temp_Path & "\thash.csv"
    Dim thFile As New System.IO.StreamWriter(tempHashFile)
    Dim outFile As New System.IO.StreamWriter(csvFile)
    'Formatting as needed
    Dim proModule As ProcessModule
    Dim comLine As String = ""
    Dim sPath As String
    Dim i As Integer


    For Each oProc As Process In oProcesses
        If Not (oProc.Id = 0) And Not (oProc.Id = 4) Then
            Dim Parent_ID As Integer
            Dim objWMI, objProc, resQuery
            'Creating a WMI object
            'impersonate means that the current user's permissions will be used by WMI
            'In a nutshell, connecting to WMI using defaults
            objWMI =
GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2")
            resQuery = objWMI.ExecQuery("Select * from Win32_Process where
ProcessID=" & oProc.Id)
```


© 2013 The SANS Institute Author retains full rights.

```
            For Each objProc In resQuery
               'For each process, get the parent ID
               Parent_ID = objProc.ParentProcessID()
               ' To handle the DBNULL to string conversion exception, add a space
               comLine = objProc.CommandLine() & Space(1)
            Next
            Try
               'For each process, get the module names
               Dim args(1) As Object
               Dim ms As New ManagementObjectSearcher("SELECT * FROM
Win32_Process WHERE ProcessId = " & oProc.Id)
               Dim usname As String
               For Each mo As ManagementObject In ms.Get
                  If CUInt(mo.InvokeMethod("GetOwner", args)) = 0 Then
                     usname = args(1).ToString & ": " & args(0).ToString
                     sPath = oProc.MainModule.FileName
                     Dim proStartInfo As New ProcessStartInfo(oProc.ProcessName)
                     oProc.StartInfo = proStartInfo
                     Dim proModuleCollection As ProcessModuleCollection
                     proModuleCollection = oProc.Modules
                     Dim UCFilename As String
                     For i = 0 To proModuleCollection.Count() - 1
                        proModule = proModuleCollection(i)
                        Proc_sql(mydate, PC_ID, oProc.ProcessName,
oProc.HandleCount.ToString, sPath, oProc.Id.ToString, oProc.Threads.Count.ToString,
comLine, Parent_ID.ToString, proModule.FileName, usname)
                        'Changing case to maintain consistency
                        UCFilename = UCase(proModule.FileName.ToString)
                        thFile.WriteLine(UCFilename)
                     Next i

                  End If
               Next
            Catch ex As Exception
               error_sql("Error in process function:" & ex.Message)
            End Try
         End If
      Next
      outFile.Close()
      thFile.Close()
      Dim tlines As String() = IO.File.ReadAllLines(Temp_Path & "\thash.csv")
      'Getting the distinct names of loaded modules (to avoid recalculation of
      'hash for the same file.
      Dim distinctlines As String() = tlines.Distinct.ToArray()
      IO.File.WriteAllLines(Temp_Path & "\hash.csv", distinctlines)
      Dim hlines As String() = IO.File.ReadAllLines(Temp_Path & "\hash.csv")
```


© 2013 The SANS Institute Author retains full rights.

```
        'Formatting as needed
        For Each hline In hline s

            'XP does raises error on some characters in filenames while Win 7 ignores them
            'So remove invalid characters altogether from filenames
            'If no invalid characters present, no harm done
            hline = hline.Trim(Path.GetInvalidFileNameChars())
            hline = hline.Trim(Path.GetInvalidPathChars())
            Dim theFile As New FileInfo(hline)
            Dim theFileVInfo As FileVersionInfo
            theFileVInfo = FileVersionInfo.GetVersionInfo(hline)
            MD5_sql(mydate, PC_ID, hline, MD5CalcFile(hline),
theFile.LastWriteTime.ToString("yyyy-MM-dd HH:mm:ss"),
theFile.LastAccessTime.ToString("yyyy-MM-dd HH:mm:ss"),
theFile.CreationTime.ToString("yyyy-MM-dd HH:mm:ss"),
theFileVInfo.CompanyName, theFileVInfo.FileVersion)
        Next
        md5writer.Close()
        System.IO.File.Delete(tempHashFile)
        System.IO.File.Delete(Temp_Path & "\hash.csv")
        Return Err()

    End Function

    '*****************************************************************
    'Get the PC Information
    '*****************************************************************
    Function PCInform()

        Dim ip() As System.Net.IPAddress =
System.Net.Dns.GetHostAddresses(System.Net.Dns.GetHostName())
        Dim ipadd As String = ""
        Dim mac As String = ""
        Dim Wmi As New System.Management.ManagementObjectSearcher("SELECT *
FROM Win32_NetworkAdapterConfiguration")
        'Windows 7 returns IP addresses differently than Windows XP
        'Collecting IP addresses based on the OS
        If (My.Computer.Info.OSFullName.Contains("Windows 7")) Then
            Try
                ipadd = ip(2).ToString
            Catch ex As Exception
                ipadd = ip(1).ToString
            End Try
        ElseIf My.Computer.Info.OSFullName.Contains("Windows XP") Then
            ipadd = ip(0).ToString
        End If
```


© 2013 The SANS Institute Author retains full rights.

```
    For Each WmiObj As ManagementObject In Wmi.Get
        If CBool(WmiObj("IPEnabled")) Then
            If (WmiObj("IPAddress")(0)) = ipadd Then
                mac = WmiObj("MACAddress")
            End If
        End If
    Next
    '
    'Determine What users is currently logged into the system by getting the user that is
running explorer.exe shell
    Dim ActualUserName As String = ""
    Dim CurrentProcesses As Management.ManagementObjectCollection
    Dim ProcessSearch As Management.ManagementObjectSearcher
    Dim ProcessItem As Management.ManagementObject
    ProcessSearch = New Management.ManagementObjectSearcher("Select * from
Win32_Process")
    CurrentProcesses = ProcessSearch.Get
    For Each ProcessItem In CurrentProcesses
        Dim ProcessOwner(2) As String
        ProcessItem.InvokeMethod("GetOwner", ProcessOwner)
        If (ProcessItem("Name").ToString = "explorer.exe") Then
            ActualUserName = ProcessOwner(0).ToString
            LocalUserName = ProcessOwner(0).ToString
            '  Console.WriteLine(ActualUserName)
            Exit For
        End If
    Next
    '
    'Determine if the Users logged in is part of the local computer admin user group
    Dim serverName = "."
    Dim oGroup As Object = GetObject("WinNT://" & serverName &
"/Administrators") 'Get list of local users that are part of admin group
    Dim isadmin As String = ""
    If Err.Number = 0 Then
        Dim bUserExist As Boolean = False ' init value
        Dim oUser As Object

        'Console.WriteLine("COMPUTER: " & serverName)
        'Console.WriteLine("************************************")

        For Each oUser In oGroup.Members
            ' Console.WriteLine(oUser.Name)
            If ActualUserName = oUser.Name Then ' Both Logged in user and admin
group user are the same
                isadmin = "Y"
                ' Console.WriteLine(isadmin)
```


© 2013 The SANS Institute Author retains full rights.

```
            Exit For    'If yes exit for loop
         Else
            isadmin = "N"
            ' Console.WriteLine(isadmin)
         End If
      Next

   End If

   'Getting the OS Architecture - 32 or 64 bit
   osArch = IntPtr.Size * 8


   Dim useraccounts As String = Temp_Path & "\ua.csv"
   Dim uaWriter As New System.IO.StreamWriter(useraccounts)
   Dim objWshNet As Object = CreateObject("WScript.Network")
   Dim strComputer As String = objWshNet.ComputerName ' local computer
   Dim objWMIService, colItems, obj
   objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\cimv2")
   colItems = objWMIService.ExecQuery _
   ("Select * from Win32_UserAccount Where Domain = '" & strComputer & "'")
   For Each obj In colItems
      If obj.Disabled = "False" Then
         uaWriter.WriteLine(obj.Caption)
      End If
   Next
   uaWriter.Close()

   'Listing all enabled users
   Dim readusers As String() = IO.File.ReadAllLines(useraccounts)
   Dim readuser As String
   Dim mystring As String = ""
   For Each readuser In readusers
      mystring = mystring + readuser + " "
   Next
   PCinfo_sql(PC_ID, mydate, Machine, ipadd, mac, My.Computer.Info.OSFullName,
System.Environment.OSVersion.ToString, osArch + "bit", ActualUserName, isadmin,
mystring)
   Return Err()

End Function

'*******************************************************************
'Query Port 3306 on selected server to see if it is available.
'*******************************************************************
Function QueryPort()
```


© 2013 The SANS Institute Author retains full rights.

```
    Dim servip = System.Net.IPAddress.Parse(server)
    Dim hostadd As System.Net.IPAddress = servip 'users server from SQL server ip
declaired at top
    Dim EPhost As New System.Net.IPEndPoint(hostadd, 3306)
    Dim s As New
System.Net.Sockets.Socket(System.Net.Sockets.AddressFamily.InterNetwork,
System.Net.Sockets.SocketType.Stream, System.Net.Sockets.ProtocolType.Tcp)
    Try
      s.Connect(EPhost)
    Catch
    End Try
    If Not s.Connected Then
            QueryPort = False

      '****************Write TO Eventlog
      Dim sSource As String
      Dim sLog As String
      Dim sEvent As String
      Dim sMachine As String

      sSource = "OHIDS"
      sLog = "Application"
      sEvent = "Network Connection Failed to Database Server"
      sMachine = "."


      Dim ELog As New EventLog(sLog, sMachine, sSource)
      ELog.WriteEntry(sEvent)
      ELog.WriteEntry(sEvent, EventLogEntryType.Warning, 234, CType(3, Short))

    Else
      QueryPort = True
    End If

  End Function

  '*********************************************************************
  'RegRead function
  '*********************************************************************
  Function ReadRegistry(ByVal strHive, ByVal strKeyPath, ByVal strValueName,
ByVal strValue)
    Dim readval As String
    On Error Resume Next
    Err.Clear()
    'oReg.GetStringValue(strHive, strKeyPath, strValueName, strValue)
```


© 2013 The SANS Institute Author retains full rights.

```
        If Err.Number = 0 Then
            ReadRegistry = My.Computer.Registry.GetValue("HKEY_LOCAL_MACHINE"
& Registery_Path, "LastFindSSN", Nothing)
        Else
            ReadRegistry = False
        End If
        Return ReadRegistry
    End Function
    '*****************************************************************
    'RegWrite function
    '*****************************************************************
    Function WriteRegistry(ByVal strKey, ByVal strValue, ByVal strRegType)
        Try
            Return oShell.RegWrite(strKey, strValue, strRegType)
        Catch ex As Exception
            error_sql("Error writing registery:" & ex.Message)
        End Try
        Return vbNullString 'stops vb.net error
    End Function
    '*****************************************************************
    ' Specify the path to a file and this routine will calculate your hash
    '*****************************************************************
    Public Function MD5CalcFile(ByVal filepath As String) As String
        ' open file (as read-only)
        Using reader As New System.IO.FileStream(filepath, IO.FileMode.Open,
IO.FileAccess.Read)
            Using md5 As New System.Security.Cryptography.MD5CryptoServiceProvider
                ' hash contents of this stream
                Dim hash() As Byte = md5.ComputeHash(reader)
                ' return formatted hash
                Return ByteArrayToString(hash)
            End Using
        End Using
    End Function
    '*****************************************************************
    ' Utility function to convert a byte array into a hex string
    '*****************************************************************
    Private Function ByteArrayToString(ByVal arrInput() As Byte) As String
        Dim sb As New System.Text.StringBuilder(arrInput.Length * 2)
        For i As Integer = 0 To arrInput.Length - 1
            sb.Append(arrInput(i).ToString("X2"))
        Next
        Return sb.ToString().ToLower
    End Function
    Private Function SplitFields(ByVal s As String) As String()
        Return Regex.Split(s, "\s+")
```


© 2013 The SANS Institute Author retains full rights.

```
End Function

Public Sub MySQLOpenConnection(ByVal pConnectionString As String)
   MysqlConn = New MySqlConnection()
   MysqlConn.ConnectionString = pConnectionString
   Try
      Dim Insert As New MySqlCommand
      MysqlConn.Open()
   Catch myerror As MySqlException
      '   Console.Write("Cannot connect to database: " & myerror.Message)
      ' Console.Read() ' Pauses the box for errors

      '****************Write TO Eventlog
      Dim sSource As String
      Dim sLog As String
      Dim sEvent As String
      Dim sMachine As String

      sSource = "OHIDS"
      sLog = "Application"
      sEvent = "Failed to Authenicate the Database Server"
      sMachine = "."


      Dim ELog As New EventLog(sLog, sMachine, sSource)
      ELog.WriteEntry(sEvent)
      ELog.WriteEntry(sEvent, EventLogEntryType.Warning, 234, CType(3, Short))


   End Try
End Sub

Public Sub MySQLCloseConnection()
   'Dim MysqlConn As MySqlConnection
   Try
      If Not MysqlConn Is Nothing Then
         If MysqlConn.State = ConnectionState.Open Then
            MysqlConn.Close()
            MysqlConn.Dispose()
         End If
      End If
   Catch ex As MySqlException
      error_sql("Error chould not close connection:" & ex.Message)
   End Try
End Sub
```

Tom Webb, tcw3bb@gmail.com

© 2013 The SANS Institute Author retains full rights.

```
   Function netstat_sql(ByVal Ndate As String, ByVal Ncname As String, ByVal
Nprotocol As String, ByVal Nlocaladdress As String, ByVal Nforeignaddress As String,
ByVal Nstate As String, ByVal Npid As String)
      Dim Insert As New MySqlCommand
      Try
         Insert.Connection = MysqlConn
         Insert.CommandText = "Insert into Netstat_Temp
(UID,Date,PC_Id,Protocol,LocalIP,DstIP,Status,PID)" & "VALUES (0," & Chr(34) &
Ndate & Chr(34) & "," & Chr(34) & Ncname & Chr(34) & "," & Chr(34) & Nprotocol &
Chr(34) & "," & Chr(34) & Nlocaladdress & Chr(34) & "," & Chr(34) &
Nforeignaddress & Chr(34) & "," & Chr(34) & Nstate & Chr(34) & "," & Chr(34) &
Npid & Chr(34) & ")"
         ' Console.Write(Insert.CommandText)
         'Console.Read()
         Insert.ExecuteNonQuery() 'runs the query Insert
      Catch ex As MySqlException
         error_sql("Error in netstat_sql:" & ex.Message)
      End Try
      Return vbNullString 'stops vb.net error
   End Function

   Function firewall_sql(ByVal FWdate As String, ByVal FWcname As String, ByVal
FWpolicy As String, ByVal FWmode As String, ByVal FWprog_name As String, ByVal
FWprog_path As String, ByVal FWtraffic_dir As String)
      Dim Insert As New MySqlCommand
      Try
         Insert.Connection = MysqlConn
         Dim command As String = "Insert into Firewall_Temp
(UID,Date,PC_Id,Policy,Mode,Prog_Name,Prog_Path,Traffic_Dir)" & "VALUES (0," &
Chr(34) & FWdate & Chr(34) & "," & Chr(34) & FWcname & Chr(34) & "," & Chr(34)
& FWpolicy & Chr(34) & "," & Chr(34) & FWmode & Chr(34) & "," & Chr(34) &
FWprog_name & Chr(34) & "," & Chr(34) & FWprog_path & Chr(34) & "," & Chr(34)
& FWtraffic_dir & Chr(34) & ")"
         Dim command_replace = Regex.Replace(command, "\\", "/")
         Insert.CommandText = command_replace
         'Console.Write(Insert.CommandText)
         ' Console.Read()
         Insert.ExecuteNonQuery() 'runs the query Insert
      Catch ex As MySqlException
         error_sql("Error in firewall_sql:" & ex.Message)
      End Try
      Return vbNullString 'stops vb.net error
   End Function
```


© 2013 The SANS Institute Author retains full rights.

```
   Function schtask_sql(ByVal STdate As String, ByVal STcname As String, ByVal
STtask_name As String, ByVal STtask_run As String, ByVal STnext_time_run As
String, ByVal STlast_time_run As String, ByVal STauthor As String, ByVal STstate As
String, ByVal STtype As String, ByVal STrun_as As String)
      Dim Insert As New MySqlCommand
      Try
         Insert.Connection = MysqlConn
         Dim command As String = "Insert into Sch_Tasks_Temp
(UID,Date,PC_Id,Task_Name,Task_Run,Next_Run_Time,Last_Run_Time,Author,State,
Type,Run_As)" & "VALUES (0," & Chr(34) & STdate & Chr(34) & "," & Chr(34) &
STcname & Chr(34) & "," & Chr(34) & STtask_name & Chr(34) & "," & Chr(34) &
STtask_run & Chr(34) & "," & Chr(34) & STnext_time_run & Chr(34) & "," & Chr(34)
& STlast_time_run & Chr(34) & "," & Chr(34) & STauthor & Chr(34) & "," & Chr(34)
& STstate & Chr(34) & "," & Chr(34) & STtype & Chr(34) & "," & Chr(34) & STrun_as
& Chr(34) & ")"
         Dim command_replace = Regex.Replace(command, "\\", "/")
         Insert.CommandText = command_replace
         ' Console.Write(Insert.CommandText)
         ' Console.Read()
         Insert.ExecuteNonQuery() 'runs the query Insert
      Catch ex As MySqlException
         error_sql("Error in schtask_sql:" & ex.Message)
      End Try
      Return vbNullString 'stops vb.net error
   End Function

   Function start_list_sql(ByVal SLdate As String, ByVal SLname As String, ByVal
SLcaption As Object, ByVal SLcommand As Object, ByVal SLdescription As Object,
ByVal SLlocation As Object, ByVal SLSettingID As Object, ByVal SLuser As Object)
      Dim Insert As New MySqlCommand
      Dim SLcommand_nq = Regex.Replace(SLcommand, """", "") ' remove quote

      Try
         Insert.Connection = MysqlConn
         Dim command As String = "Insert into Start_List_Temp
(UID,Date,PC_Id,Caption,Command,Description,Location,SettingID,User)" &
"VALUES (0," & Chr(34) & SLdate & Chr(34) & "," & Chr(34) & SLname & Chr(34) &
"," & Chr(34) & SLcaption & Chr(34) & "," & Chr(34) & SLcommand_nq & Chr(34) &
"," & Chr(34) & SLdescription & Chr(34) & "," & Chr(34) & SLlocation & Chr(34) &
"," & Chr(34) & SLSettingID & Chr(34) & "," & Chr(34) & SLuser & Chr(34) & ")"
         Dim command_replace = Regex.Replace(command, "\\", "/") ' this changed the
back slash to forward slash
         Insert.CommandText = command_replace ' set variable to run
         ' Console.Write(Insert.CommandText)
         ' Console.Read()
         Insert.ExecuteNonQuery() 'runs the query Insert
```


© 2013 The SANS Institute Author retains full rights.

```
        Catch ex As MySqlException
            error_sql("Error in start_list_sql:" & ex.Message)
        End Try
        Return vbNullString 'stops vb.net error
    End Function


    Function service_list_sql(ByVal Svdate As String, ByVal Svname As Object, ByVal
Svpid As Object, ByVal Svpathname As Object, ByVal Svstartmode As Object, ByVal
Svstate As Object, ByVal Svstatus As Object, ByVal Svtype As Object, ByVal
Svstartname As Object, ByVal Svexitcode As Object)
        Dim Insert As New MySqlCommand
        Dim Svpathname_nq = Regex.Replace(Svpathname, """", "") ' remove quote
        ' Console.Write(Svpathname_nq)
        '  Console.Read()
        Try
            Insert.Connection = MysqlConn
            Dim command As String = "Insert into Service_List_Temp
(UID,Date,PC_Id,Name,PID,PathName,StartMode,State,Status,ServiceType,StartName,
Exitcode)" & "VALUES (0," & Chr(34) & Svdate & Chr(34) & "," & Chr(34) & PC_ID
& Chr(34) & "," & Chr(34) & Svname & Chr(34) & "," & Chr(34) & Svpid & Chr(34) &
"," & Chr(34) & Svpathname_nq & Chr(34) & "," & Chr(34) & Svstartmode & Chr(34)
& "," & Chr(34) & Svstate & Chr(34) & "," & Chr(34) & Svstatus & Chr(34) & "," &
Chr(34) & Svtype & Chr(34) & "," & Chr(34) & Svstartname & Chr(34) & "," & Chr(34)
& Svexitcode & Chr(34) & ")"
            Dim command_replace = Regex.Replace(command, "\\", "/") ' this changed the
back slash to forward slash
            '  Console.WriteLine(command_replace)
            Insert.CommandText = command_replace ' set variable to run
            ' Console.Write(Insert.CommandText)
            ' Console.Read()
            Insert.ExecuteNonQuery() 'runs the query Insert
        Catch ex As MySqlException
            error_sql("Error in service_list_sql:" & ex.Message)
        End Try
        Return vbNullString 'stops vb.net error
    End Function


    Function Proc_sql(ByVal Pdate As String, ByVal Pname As String, ByVal Pprocname
As Object, ByVal Phandlecount As Object, ByVal Pprocfile As Object, ByVal Ppid As
Object, ByVal Pthreadcount As Object, ByVal Pcommandline As Object, ByVal Pppid
As Object, ByVal Pprocmod As Object, ByVal Powner As Object)
        Dim Insert As New MySqlCommand
        Pcommandline = Regex.Replace(Pcommandline, """", "") ' remove quote
```


© 2013 The SANS Institute Author retains full rights.

```
        Try
            Insert.Connection = MysqlConn
            Dim command As String = "Insert into Process_Temp
(UID,Date,PC_Id,Proc_Name,Handelcount,Proc_File,PID,Threadcount,Commandline,PP
ID,Proc_Mod,Owner)" & "VALUES (0," & Chr(34) & Pdate & Chr(34) & "," & Chr(34)
& Pname & Chr(34) & "," & Chr(34) & Pprocname & Chr(34) & "," & Chr(34) &
Phandlecount & Chr(34) & "," & Chr(34) & Pprocfile & Chr(34) & "," & Chr(34) & Ppid
& Chr(34) & "," & Chr(34) & Pthreadcount & Chr(34) & "," & Chr(34) &
Pcommandline & Chr(34) & "," & Chr(34) & Pppid & Chr(34) & "," & Chr(34) &
Pprocmod & Chr(34) & "," & Chr(34) & Powner & Chr(34) & ")"
            Dim command_replace = Regex.Replace(command, "\\", "/") ' this changed the
back slash to forward slash
            Insert.CommandText = command_replace ' set variable to run
            'Console.Write(Insert.CommandText)
            'Console.Read()
            Insert.ExecuteNonQuery() 'runs the query Insert
        Catch ex As MySqlException
            error_sql("Error in Proc_sql:" & ex.Message)
        End Try
        Return vbNullString 'stops vb.net error
    End Function


    Function MD5_sql(ByVal Hdate As String, ByVal Hname As String, ByVal
Hfilename As Object, ByVal Hmd5 As Object, ByVal Hmdate As Object, ByVal Hadate
As Object, ByVal Hcdate As Object, ByVal Hcompanyname As Object, ByVal Hversion
As Object)
        Dim Insert As New MySqlCommand
        '   Pcommandline = Regex.Replace(Pcommandline, """", "") ' remove quote

        Try
            Insert.Connection = MysqlConn
            Dim command As String = "Insert into PC_Hash_Temp
(UID,Date,PC_Id,File_Name,MD5,Mdate,Adate,Cdate,Company_Name,Version)" &
"VALUES (0," & Chr(34) & Hdate & Chr(34) & "," & Chr(34) & Hname & Chr(34) &
"," & Chr(34) & Hfilename & Chr(34) & "," & Chr(34) & Hmd5 & Chr(34) & "," &
Chr(34) & Hmdate & Chr(34) & "," & Chr(34) & Hadate & Chr(34) & "," & Chr(34) &
Hcdate & Chr(34) & "," & Chr(34) & Hcompanyname & Chr(34) & "," & Chr(34) &
Hversion & Chr(34) & ")"
            Dim command_replace = Regex.Replace(command, "\\", "/") ' this changed the
back slash to forward slash
            Insert.CommandText = command_replace ' set variable to run
            ' Console.Write(Insert.CommandText)
            'Console.Read()
            Insert.ExecuteNonQuery() 'runs the query Insert
        Catch Ex As MySqlException
```


© 2013 The SANS Institute Author retains full rights.

```
        error_sql("Error in MD5_sql:" & Ex.Message)
    End Try
    Return vbNullString 'stops vb.net error
End Function


Function comid_sql(ByVal Compname As Object)

    Dim Q As New MySqlCommand
    Try
        Dim myname As MySqlParameter
        Dim PCID As MySqlParameter
        Dim PCIDreader As MySqlDataReader
        Q = New MySqlCommand("get_com_id", MysqlConn)
        Q.CommandType = CommandType.StoredProcedure
        myname = Q.Parameters.Add("Compname_id", MySqlDbType.Text)
        PCID = Q.Parameters.Add("out", MySqlDbType.Text)
        PCID.Direction = ParameterDirection.Output
        myname.Value = Compname
        PCIDreader = Q.ExecuteReader()

        While PCIDreader.Read()
            PC_ID = PCIDreader.GetString(0)
            '   Console.WriteLine(PC_ID)
            '  Console.Read()
        End While
        PCIDreader.Close()
    Catch ex As MySqlException
        error_sql("Error in comid_sql:" & ex.Message)
    End Try

    Return vbNullString 'stops vb.net error
End Function

Function PCinfo_sql(ByVal PpcID As Integer, ByVal Pcdate As String, ByVal
Pcname As String, ByVal Pcip As String, ByVal Pcmac As String, ByVal Pcosname As
Object, ByVal Pcosver As String, ByVal PCarch As String, ByVal Pcuser As Object,
ByVal Pcadmin As Object, ByVal Pcenabledusers As String)

    Dim Update As New MySqlCommand
    '   Pcommandline = Regex.Replace(Pcommandline, """", "") ' remove quote
    Try
        Update = New MySqlCommand("update_comp_info", MysqlConn)
        Update.CommandType = CommandType.StoredProcedure
        Dim Auth_Key As String = getMD5Hash(Machine & PC_ID) ' this is used to
prevenet people from guessing and overwriting records
```


© 2013 The SANS Institute Author retains full rights.

```
        Update.Parameters.Add("Auth_Key", MySqlDbType.Text)
        Update.Parameters.Add("PpcID", MySqlDbType.Text)
        Update.Parameters.Add("Pcdate", MySqlDbType.Text)
        Update.Parameters.Add("Pcip", MySqlDbType.Text)
        Update.Parameters.Add("Pcmac", MySqlDbType.Text)
        Update.Parameters.Add("Pcosname", MySqlDbType.Text)
        Update.Parameters.Add("Pcosver", MySqlDbType.Text)
        Update.Parameters.Add("Pcarch", MySqlDbType.Text)
        Update.Parameters.Add("Pcuser", MySqlDbType.Text)
        Update.Parameters.Add("Pcadmin", MySqlDbType.Text)
        Update.Parameters.Add("Pcenabledusers", MySqlDbType.Text)
        '
        Update.Parameters("Auth_Key").Value = Auth_Key
        Update.Parameters("PpcID").Value = PpcID
        Update.Parameters("Pcdate").Value = Pcdate
        Update.Parameters("Pcip").Value = Pcip
        Update.Parameters("Pcmac").Value = Pcmac
        Update.Parameters("Pcosname").Value = Pcosname
        Update.Parameters("Pcosver").Value = Pcosver
        Update.Parameters("Pcarch").Value = PCarch
        Update.Parameters("Pcuser").Value = Pcuser
        Update.Parameters("Pcadmin").Value = Pcadmin
        Update.Parameters("Pcenabledusers").Value = Pcenabledusers
        Update.ExecuteNonQuery()


    Catch ex As MySqlException
        error_sql("Error in PCINfo_sql:" & ex.Message)
    End Try
    Return vbNullString 'stops vb.net error



End Function
Function getMD5Hash(ByVal strToHash As String) As String
    Dim md5Obj As New Security.Cryptography.MD5CryptoServiceProvider
    Dim bytesToHash() As Byte = System.Text.Encoding.ASCII.GetBytes(strToHash)

    bytesToHash = md5Obj.ComputeHash(bytesToHash)

    Dim strResult As String = ""

    For Each b As Byte In bytesToHash
        strResult += b.ToString("x2")
    Next
```


© 2013 The SANS Institute Author retains full rights.

```
        Return strResult
    End Function

    Public Function text_combine( _
    ByVal path_to_read_file As String, _
    ByVal path_to_append_file As String _
) As Boolean

        'Console.Write(path_to_append_file)
        'Console.Read()
        If ( _
            (IO.File.Exists(path_to_read_file)) _
            And (IO.File.Exists(path_to_append_file)) _
        ) Then
            Try
                System.IO.File.AppendAllText( _
                    path_to_append_file, _
                    System.IO.File.ReadAllText(path_to_read_file) _
                )
                text_combine = True
            Catch ex As Exception
                text_combine = False
                error_sql("Error in text_combine:" & ex.Message)
            End Try
        Else
            text_combine = False
        End If

    End Function

    Function findssn_sql(ByVal ssndate As String, ByVal ssncount As Integer, ByVal
ssnfile As String)
        Dim Insert As New MySqlCommand
        Try
            Insert.Connection = MysqlConn
            Dim command As String = "Insert into Find_SSN_Temp
(UID,Date,PC_Id,Count,File)" & "VALUES (0," & Chr(34) & ssndate & Chr(34) & "," 
& Chr(34) & PC_ID & Chr(34) & "," & Chr(34) & ssncount & Chr(34) & "," & Chr(34)
& ssnfile & Chr(34) & ")"
            Dim command_replace = Regex.Replace(command, "\\", "/") ' this changed the
back slash to forward slash
            Insert.CommandText = command_replace ' set variable to run

            'Console.Write(Insert.CommandText)
            ' Console.Read()
            Insert.ExecuteNonQuery() 'runs the query Insert
```


© 2013 The SANS Institute Author retains full rights.

```
        Catch ex As MySqlException
            error_sql("Error in findssn_sql:" & ex.Message)
        End Try
        Return vbNullString 'stops vb.net error
    End Function

    Function error_sql(ByVal strError As String) ' input errors into the SQL database for
troubleshooting
        Dim strError_nq = Regex.Replace(strError, """", "") ' remove quote
        Dim Insert As New MySqlCommand
        Try
            Insert.Connection = MysqlConn
            Dim command As String = "Insert into Error_Log (UID,Date,PC_Id,Error)" &
"VALUES (0," & Chr(34) & Format(System.DateTime.Now, "yyyy-MM-dd
HH:mm:ss") & Chr(34) & "," & Chr(34) & PC_ID & Chr(34) & "," & Chr(34) &
strError_nq & Chr(34) & ")"
            Dim command_replace = Regex.Replace(command, "\\", "/") ' this changed the
back slash to forward slash
            Insert.CommandText = command_replace ' set variable to run
            'Console.Write(Insert.CommandText)
            'Console.Read()
            Insert.ExecuteNonQuery() 'runs the query Insert
        Catch ex As MySqlException
            'Console.Write("database insert error for errorlog : " & ex.Message)
            'Console.Read()
        End Try
        Return vbNullString 'stops vb.net error
        Return vbNullString 'stops vb.net error
    End Function



End Module
```

Appendix E ohids-report.sh

Tom Webb, tcw3bb@gmail.com

© 2013 The SANS Institute Author retains full rights.

```
#!/bin/bash
#0.1
#OHIDS REPORTING ENGINE
#Tom Webb
#Latest version of the code is available at https://code.google.com/p/open-source-host-based-ids/


WORKINGDIR=$(/bin/mktemp -d)
DATE=`date -d "-0 day" +%F`
cd $WORKINGDIR
DB_USER=client
DB_NAME=OHIDS
PC_ID=$3
if [ $# -eq 0 ]; then
	echo
	echo "Usage: $0 -t type "
	echo ""
	echo "By Default the TEMP table will be used for most recent data"
	echo "Common reports:"
	echo "-t Proc_Odd Get processes that are odd. Also has AV  hash results"
	echo "-t Proc_Loc Get processes run from temp directories."
	echo "-t Proc_Diff Shows new Processes running different from previous day"
	echo "-t Proc_Date Shows Files running that have a modified or create date in that
last 48 hours"
	echo "-t Start_Diff New items in Computers startup list from the previous day"
	echo "-t Start_Loc shows process in startup in temp directories"
	echo "-t Service_Diff new services on Computers compared to previous day"
	echo "-t Hash_Comp compare hashs of exe to version numbers"
	echo "-t Firewall_Diff Shows firewall changes between different days"
	echo "-t SSN_Top will display top 25 highest SSN Count per Computer"
	echo "-t SSN_Comp will display the top 50 file for a give PC_ID"
	exit
fi

case $1 in
#-t) type=$2; shift 2;;
-t) type=$2;
esac



query_proc_loc()
{
echo "select DISTINCT PC_Id,Proc_File from Process_Temp where Proc_File not like
'%Program Files%' and  Proc_File not like '%system32%' and Proc_File not like
```


© 2013 The SANS Institute Author retains full rights.

```
'%sysWow64%' and Proc_File not like '%windows%' and Proc_File not like
'%PROGRA%' and Proc_File not like '%Google% and Proc_File not like in (select
DISTINCT Name from Good_File)';" >sql.statement

mysql -u $DB_USER $DB_NAME < sql.statement | sed 's/\t/","/g;s/^/"/;s/$/"/;s/\n//g'>
results.csv

cat results.csv
}


query_proc_odd()
{
echo "select DISTINCT PC_Id,Proc_File from Process_Temp where Proc_File like
'%recyler%' or  Proc_File like '%system volume information%' or Proc_File like
'%temp%' or Proc_File like '%tmp%' and Proc_File not in (select DISTINCT Name from
Good_File);" >sql.statement

mysql  -u $DB_USER $DB_NAME < sql.statement | sed 's/\t/","/g;s/^/"/;s/$/"/;s/\n//g' >
results.csv

while  IFS=',' read ID File
do
       echo "select PC_Id,File_Name,MD5 from PC_Hash_Temp where PC_Id=$ID and
File_Name=$File;" >hash.sql
       mysql --skip-column-names -u $DB_USER $DB_NAME  < hash.sql  | sed
's/\t/,/g;s/\n//g' >> hash.csv
done <results.csv


#Prep file for bulk load
echo "begin" >mal-hash
cut -d ',' -f3 hash.csv |sort |uniq >>mal-hash #TRIM DOWN TO UNIQ RESULTS TO
PLAY NICE
echo "end" >>mal-hash
netcat hash.cymru.com 43 <mal-hash >malhash-result
grep -v '#' malhash-result >mal-filtered

while  IFS=',' read  ID File Hash
do
 av_result=`grep -m1 $Hash mal-filtered`
 echo $ID $File $av_result >>final
done <hash.csv

cat final
```

Tom Webb, tcw3bb@gmail.com

© 2013 The SANS Institute Author retains full rights.

```
}

query_start_diff()
{
#determine last PCID
mysql -B --skip-column-names -u $DB_USER $DB_NAME -e "SELECT DISTINCT
PC_Id from Start_List_Temp;" >pcid #Get PC_Ids from previous day PC's

while read i
do  #FOR Each PC GET A DIFF

mysql  -B -u $DB_USER $DB_NAME -e "SELECT PC_Info.PC_Id, Cname ,Command
from Start_List_Temp, PC_Info where PC_Info.PC_Id=$i and PC_Info.PC_Id =
Start_List_Temp.PC_Id and PC_Info.Last_Seen != PC_Info.First_Seen and Command
not in (select DISTINCT Command from Start_List where PC_ID=$i);"

done<pcid
}

query_service_diff()
{
mysql -B --skip-column-names -u $DB_USER $DB_NAME -e "SELECT DISTINCT
PC_Id from Service_Temp;" >pcid #Get PC_Ids from previous day PC's
while read i
  do  #FOR Each PC GET A DIFF

mysql -B -u $DB_USER $DB_NAME  -e "SELECT PC_Info.PC_Id, Cname ,Name
from Service_List_Temp, PC_Info where PC_Info.PC_Id=$i and PC_Info.PC_Id =
Service_List_Temp.PC_Id and PC_Info.Last_Seen != PC_Info.First_Seen and Name not
in (select DISTINCT Name from Service_List where PC_ID=$i)";
done<pcid
}


query_proc_diff()
{
mysql -B --skip-column-names -u $DB_USER $DB_NAME -e "SELECT DISTINCT
PC_Id from Process_Temp;" >pcid #Get PC_Ids from previous day PC's
while read i
 do
mysql -B -u $DB_USER $DB_NAME  -e "SELECT DISTINCT PC_Id, Proc_File from
Process where PC_Id=$i and Proc_File not in (select DISTINCT Proc_File from
Process_Temp where PC_ID=$i);"
done<pcid
}
```

Tom Webb, tcw3bb@gmail.com

© 2013 The SANS Institute Author retains full rights.

```
query_proc_date()
{
qdate=`date -d "2 day ago" +%Y-%m-%d`
mysql -B -u $DB_USER $DB_NAME -e "SELECT PC_Id, File_Name,MD5 FROM
PC_Hash_Temp where CDATE >='$qdate 00:00:00'OR MDATE >='$qdate 00:00:00'
order by MD5;"

}

query_start_loc()
{
mysql -B -u $DB_USER $DB_NAME -e "select DISTINCT PC_Id,Command from
Start_List_Temp where Command not like '%Program Files%' and Command not like
'%/AppData/Local/Google%' and  Command not like '%system32%' and Command not
like '%sysWow64%' and Command not like '%windows%' and Command not like
'%PROGRA%' and Command not in (select DISTINCT Name from Good_File);"
}

query_hash_comp()
{
#C:/WINDOWS/SYSTEM32/NLSLEXICONS0011.DLL 6.1.7600.16385
(win7_rtm.090713-1255)   f95bef6d4afb35cacb8daf5ff1df8769

mysql -B -u slapc ITSO_PC_IR -e "select File_Name,Version,MD5 from PC_Hash
WHERE MD5 not in (select DISTINCT MD5 from Good_Hash)" >hash
cat hash |awk -F '/' '{ print $NF}'|sort |uniq >hash.sort

IFS=`printf '\n\t'`
while read file ver hash ; do

count=`awk '{ if ($1 == "'$file'" && $2 == "'$ver'" ) print $0}' hash.sort |wc -l`

if [ $count -gt 1 ];
 then
   echo $file $ver $hash
fi

done<hash.sort

}

query_firewall_diff()
{
mysql -B --skip-column-names -u $DB_USER $DB_NAME -e "SELECT DISTINCT
PC_Id from Firewall_Temp;" >pcid #Get PC_Ids from previous day PC's
```


© 2013 The SANS Institute Author retains full rights.

```
while read i
do
 mysql -B -u $DB_USER $DB_NAME  -e "SELECT PC_Info.PC_Id, Cname ,Prog_Path
from Firewall_Temp, PC_Info where PC_Info.PC_Id=$i and PC_Info.PC_Id =
Firewall_Temp.PC_Id and PC_Info.Last_Seen != PC_Info.First_Seen and Prog_Path not
in (select DISTINCT Prog_Path from Firewall where PC_ID=$i)";
done<pcid
}

query_top_ssn()
{

mysql -B -u $DB_USER $DB_NAME  -e  "select
Find_SSN.Date,Find_SSN.PC_Id,PC_Info.Cname, SUM(count) as Total, PC_Info.IP
FROM Find_SSN, PC_Info where PC_Info.PC_ID=Find_SSN.PC_Id and
Find_SSN.Date >= DATE_SUB(NOW(), INTERVAL 14 DAY ) GROUP by PC_ID
order by total desc limit 25;"

}

query_ssn_comp()
{
#PC_ID from Global $3

if [ -z $PC_ID ]; then #IF PC_ID is blank
 echo "Please enter a PC_Id"
fi

mysql -B -u $DB_USER $DB_NAME  -e  "select Count, File from Find_SSN where
PC_Id=\"$PC_ID\" and count > 50 order by count DESC;"

}


clean ()
{
rm -rf $WORKINGDIR
}


#####################################
#Determine QUERY
#####################################
if [ $type  = "Proc_Loc" ]; then #Process locaion lookup
query_proc_loc
fi
```


© 2013 The SANS Institute Author retains full rights.

```
if [ $type = "Proc_Odd" ]; then
query_proc_odd
fi
if [ $type = "Start_Diff" ]; then
query_start_diff
fi
if [ $type = "Service_Diff" ]; then
query_service_diff
fi

if [ $type = "Proc_Diff" ]; then
query_proc_diff
fi

if [ $type = "Proc_Date" ]; then
query_proc_date
fi

if [ $type = "Start_Loc" ]; then
query_start_loc
fi
if [ $type = "Hash_Comp" ]; then
query_hash_comp
fi

if [ $type = "Firewall_Diff" ]; then
query_firewall_diff
fi

if [ $type = "SSN_Top" ]; then
query_top_ssn
fi

if [ $type = "SSN_Comp" ]; then
query_ssn_comp
fi

clean
```

Tom Webb, tcw3bb@gmail.com

© 2013 The SANS Institute Author retains full rights.