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
"Advanced Incident Response, Threat Hunting, and Digital Forensics (Forensics
at http://www.giac.org/registration/gcfa

Live Response Using PowerShell

GIAC (GCFA) Gold Certification

Author: Sajeev Nair, Nair.Sajeev@gmail.com
Advisor: Antonios Atlasis

Accepted: August 7th 2013

Abstract
Live response is a critical area within Incident Response. While there are many tools and processes available to collect valuable information for later analysis, there haven't been any comprehensive studies done with the capabilities of PowerShell as an inbuilt tool to aid live response. This paper focuses on various ways in which PowerShell can be utilized to collect data from Windows 7 systems. PowerShell comes bundled with Windows 7 and Microsoft provides a wealth of options to collect, analyze and present the various artifacts.

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

1. Introduction

Organizations today handle more sensitive personal data than ever before. As the amount of sensitive personal data increases, the more they are susceptible to security incidents and breaches (AICPA, n.d). The risk also increases due to the fact that such sensitive personal data is shared with multiple entities such as clients and business partners. To mitigate this risk, organizations started investing in Incident Response programs. Having an Incident Response program allows organizations to follow a formal process while responding to security incidents (Cichonski, Millar, Grance, Scarfone, 2012).

One of the biggest challenges in Incident Response today is in the incident detection phase. Do you have the right information available to determine if a security incident has occurred? How fast can you collect the information to determine if a security incident has occurred? In this paper, various industry data breach and incident reports were studied to identify the amount of time it takes to detect the incident. According to some of these reports:

- 64% - Percentage of victim organizations that took more than 90 days to detect the intrusion. (Trustwave Global Security Report, 2013)
- 66% - In 2012, 66% of breaches remained undiscovered for months or more (Verizon Data Breach Report, 2013).
- 243 days - "median number of days that the attackers were present on a victim network before detection" (Mandiant M-Trends, 2013).

From the various reports it is apparent that organizations are struggling with incident detection. Organizations have to do a better job in detecting incidents as the incident response costs continue to increase (Ponemon Institute, 2012).

Incident identification through disk imaging and forensic analysis is very time consuming and impacts the normal operation of organization's business. Additionally, important volatile evidence could be lost by shutting down a system (Walters, Petroni,

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

2007). Due to these factors, live response is being used as a critical part in the investigation process.

There are various tools available, both open-source and commercial to perform live response. This paper focuses on a third option – use of in-built operating system tools and commands to do the job. Operating system selected for this study is Windows 7 and PowerShell is an in-built tool or scripting language that comes bundled with Windows 7. PowerShell is a very powerful and scalable scripting language using which we can extract the required information from Windows 7 operating system. This paper also looks at some of the challenges that both open source and commercial tools present for organizations.

2. Live Response

Live response is an area that deals with collecting information from a live machine in order to identify if an incident has occurred. Such data include artifacts such as process information, connection information, files opened by processes, and so on. It does not have to be only volatile information, it can be any artifact to establish the fact that an incident has occurred. Live response helps the analyst to not lose the artifacts which may not be available when the machine is powered down. This also helps an analyst to respond to an incident quickly while not disturbing the regular activity of that machine. This aspect is very important for both user machines and servers, where organizations cannot afford to have downtime until we establish the fact that an incident has occurred.

2.1. What to collect during a Live Response

The goal of live response is to identify incidents as quickly as possible. In order to do that you want to collect the right information that helps you make the decision. Here is a comprehensive list of artifacts that you want to collect (Jones, Bejtlich & Rose, 2006. Carvey, 2009. Carvey, 2011):

1. Machine and Operating system information.
2. User accounts and current login information.

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

3. Network configuration and connectivity information.
4. Anti-Virus application status and related logs.
5. Startup applications.
6. Running process related information.
7. Running services related information.
8. Drivers installed and running.
9. DLLs created.
10. Open files.
11. Open shares.
12. Mapped drives.
13. Scheduled jobs.
14. Active network connections and related process.
15. Hotfixes applied.
16. Installed applications.
17. Link files created.
18. Packed files.
19. USB related.
20. Shadow copies created.
21. Prefetch files and timestamps.
22. DNS cache.
23. List of available logs and last write times.
24. Firewall configuration.
25. Audit policy.
26. Temporary Internet files and cookies.
27. Typed URLs.
28. Important registry keys.
29. File timeline.
30. Important event logs.

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

2.2. Tools available – Commercial and Open Source

There are many tools – both open source and commercial to achieve this objective. The below list of tools is not a comprehensive list but to give the reader the products available in the market.

1. Helix3 Enterprise. This is an enterprise level solution to capture required evidences from a remote system (E-fence, n.d.a).
2. Live Response. Acquires volatile data using a USB key (E-fence, n.d.b).
3. ProDiscover Incident Response. This is an enterprise level client server application that can perform disk preview, imaging and analysis (Techpathways, n.d).
4. Mandiant for Intelligent Response. This is an appliance-based solution to investigate enterprise wide endpoints (Mandiant, n.d).
5. EnCase Enterprise. This is another enterprise level client server application, which can do multitude of incident response and forensic investigations remotely (Guidance, n.d).
6. The Windows Forensic Toolchest. This a live response tool for Windows systems (Foolmoon, n.d).
7. GRR. GRR is an Incident Response Framework focused on remote live forensics (GRR, n.d).
8. RPIER. This tool utilizes multiple open source utilities to collect artifacts from a live system. (RPIER, n.d).
9. MIR-ROR. MIR-ROR is a script that calls specific Windows sysinternals and other utilities to perform live response (MIR-ROR, n.d).

2.3. Live Response challenges

Live response is a maturing area within the incident response spectrum and there are many tools to do the job. However, relying on open source and commercial tools present many challenges:

Privacy related. Many of the tools are designed to collect information which is on the user's machine without user input. If a country's regulation requires consent to collect such information, it could pose privacy concerns.

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

Connectivity related. Current organization's internal networks are highly segmented and communication outside the segmented networks is controlled through a firewall. It is a nightmare for large organizations to manage all these connectivity in order to provide access from a centrally managed live response tool. Additionally, what happens when the tool itself changes or moved to a new network, now you have a new IP address and new application port number to deal with.

Licensing related. Do you really trust that software you are running on your network? Do you know what exactly it is supposed to do? Do you know if the software runs other tools which may be prohibitive for commercial usage? Verifying usage options for many of the open source software is a tedious task and requires legal department's approval. Even for commercial tools, you have to really understand whether it can be used on a partner network, software images which run your client's licensed operating system, personal machines that employees bring in as part of your BYOD policy, etc.

Tool installation related. Due to the memory and process footprint that the live response agents add to a user's machine, many organizations prefer an on-demand approach to installation of agents. It is also done to limit the number of licenses an organization can use. In such cases, can you make sure you can install the agent fast enough to capture the artifacts as the incidents happen? What about the artifacts left behind by the installation of such agents? During an analysis, does the agent installation come up as the last made change? Since many of the corporate users don't have administrative privilege to install and run the agents, some organizations are forced to wait for the technician to arrive and install the agent. In such situations you are not only altering the user profile but destroying vital evidence as well.

USB and CD/DVD related. Today, most organizations block USB and CD/DVD usage for users that handle sensitive information. If the tool is designed to collect artifacts using these methods, then it poses problems.

Efficiency related. If the tools are designed to collect "all or nothing", then it would not be efficient in scenarios where you are using your security intelligence and collecting specific artifacts. Specific artifacts may include items such as specific USB device connected, login time for a specific user, whether a particular process is running,

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

whether a DLL with a specific name is present, whether a specific registry entry is present, specific IP address in the active network connections, etc. Collecting specific artifacts also ensures that you are making minimal change to the system. Post confirmation of an incident, when you collect system memory, you don't want the usage of tools to overwrite memory areas.

Cost related. Commercial tools that operate on a per agent basis become very expensive for large organizations. You also need to account for operating expense such as data center cost, administrator cost, hardware cost, software cost, vendor support cost, etc.

So, what's the solution? The solution is to use in-built tools and APIs to extract artifacts. Most of the challenges discussed above can be mitigated by using in-built tools. However, the challenge is to identify if such tools can effectively collect the required artifacts in order to analyze and detect the incidents that are taking place in your network. PowerShell, which comes bundled with Windows 7, is highly customizable and can do an admirable job in collecting the required artifacts.

3. Overview of PowerShell

PowerShell is both a scripting language and a powerful interactive command interface similar to Bash in UNIX. PowerShell console where the commands are run is very similar to the Windows command interface, cmd.exe. PowerShell commands can be run in the background or interactively if a particular country's privacy policy enforces an organization to do so. PowerShell V2 is installed by default on Windows 7 operating system.

PowerShell commands or Cmdlets are based on .NET Framework objects, which mean that the objects carry multiple aspects or properties of the command. These Cmdlets lets you access the file system and other Windows operating system data stores, such as the registry. PowerShell also provides access to Windows Management Instrumentation (WMI), which means that all the WMI commands that incident responders and information security professionals are familiar with, can be run using PowerShell.

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

All the Cmdlets follow a verb-noun structure, where the verb is always an action statement to get something from the operating system or tell the operating system to do something. The noun part of the Cmdlet is what the different objects that are of interest, objects such as computer, file system, disk, processes, event logs, etc. The noun part is always derived from a specific .NET class (Microsoft, n.d.a).

All the PowerShell Cmdlets follow a common help system."Get-Help <Cmdlet name> -Online" command initiates a local internet explorer session to the Microsoft TechNet library, which contains command options with multiple examples.

Some of the most common Cmdlets that are useful for live response are:

Command	Alias	Description
Get-ChildItem	GCI or DIR or LS	Similar to "dir" command, it gets the items and child items from one or more directories. It can also identify the MAC time stamps.
Get-ItemProperty	GP	Primarily used to get the property values of registry entries.
Get-WmiObject	GWMI	Lists details of a WMI class.
Get-Process	GPS	Lists the processes that are running on the machine.
Get-Service	GSV	Lists the services that are running on the machine.
Get-WinEvent	None	Lists the events from event logs and event tracing files.
Get-HotFix	None	Lists the hotfixes applied on the machine.
Get-Content	GC	Lists the contents of a file.
Write-Host	None	Enables writing messages to the console. Useful if the command or the script need to be run interactively.

Table 3.0.1: Common PowerShell Cmdlets used in live response

Since PowerShell Cmdlets are designed to deal with objects, the output of Cmdlets carry additional data that can be used for additional processing, such as output formatting, command piping, sorting and export options. Command piping is one of the most powerful features of PowerShell. It enables outputs of one command to pass through a whole new Cmdlet and start a new set of processing. Export options include

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

HTML, Text, XML and CSV. Some of the additional processing and export options are listed below with examples:

Command	Description
Select-Object or select	Primarily used to select specific properties from a Cmdlet. Ex: Get-Process | select ID, ProcessName
Select-String	Similar to find or grep. This command can be used to select a specific string from the output. Ex: ipconfig /displaydns | select-string 'Record Name'
ConvertTo-Html	Converts the output to HTML format. This command also supports defining HEAD, TITLE and BODY options. Ex: Get-Process | select ID, ProcessName | ConvertTo-Html > c:\report.html
Format-table or ft	This is an output option to format the display in a table form. Ex: Get-Process | select ID, ProcessName | ft -auto
Where-Object or where	This command is used to filter the output for specific properties Ex: Get-WmiObject Win32_NetworkAdapterConfiguration | where{$_.IPEnabled -eq 'True'}
ConvertFrom-Csv	This command is used to convert a CSV file for formatting within PowerShell. Ex: driverquery.exe /v /FO CSV | ConvertFrom-CSV | Select 'Display Name','Start Mode', Path
Sort-Object or sort	This command sorts the properties in ascending or descending order. Ex: driverquery.exe /v /FO CSV | ConvertFrom-CSV | Select 'Display Name','Start Mode', Path | sort path
foreach-object or foreach	This is the "for" loop in the PowerShell world. Ex: Get-Process |select Modules | foreach {$_.Modules} | select Product, ModuleName
Get-Date	This command shows the current date and time. It can also be used to add or subtract days while filtering the output. Ex: Get-WinEvent -FilterHashtable @{logname='system';starttime=((Get-Date).AddDays(-1))} | Select

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

	TimeCreated,ID,Message

Table 3.0.2: PowerShell – additional processing options

PowerShell also uses multiple parameters to enhance output processing. These parameters are same for all Cmdlets. Some of the most commonly used parameters are given below with examples:

Parameter	Description
ErrorAction or EA	This command is used to specify a custom error action for each Cmdlet. The most common option is to "SilentlyContinue" or a value of "0". Ex: gci -ea 0 | select Name, LastWriteTime
Recurse or r	This is used to do an action recursively Ex: gci -recurse -ea 0 | select Name, LastWriteTime
Path	Path defines the directory to be used in the Cmdlet Ex: gci -path C:\ -recurse -ea 0 | select Name, LastWriteTime
Force	Force is used to bypass the file attribute settings of hidden and system. Ex: gci -path C:\ -recurse -force -ea 0 | select Name, LastWriteTime
Include	This parameter is used to include a specific set of files Ex: gci -path C:\ -include *.exe –r -ea 0 | select Name, LastWriteTime
max	Defines the maximum number of entries that are required in the output. Generally used with Get-WinEvent. Ex: Get-WinEvent -max 50 -ea 0 -FilterHashtable @{Logname='security';ID=4624} | ft –auto -wrap
FilterHashtable	Primarily used with Get-WinEvent Cmdlet to filter the event logs. Get-WinEvent -FilterHashtable @{Logname='security'; ID=4672} | select TimeCreated,ID,Message | ft –auto -wrap

Table 3.0.3: PowerShell – additional parameters

In PowerShell V2, there are a large number of in-built commands to satisfy the live response need. Developers can utilize the PowerShell APIs to create additional Cmdlets if required. Since PowerShell is based on .NET Framework, it also enables

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

PowerShell commands access to large collection of .NET classes. The .NET classes already provide access to various Windows system resources. Apart from these, PowerShell also lets you use the traditional tools, such as ipconfig, netstat, arp, systeminfo, openfiles, driverquery, etc.

The various Cmdlets, along with the additional processing features is what makes PowerShell really powerful. Another major advantage of using PowerShell for live response is its ability to completely automate. Automation is always beneficial as it becomes more efficient and scalable.

3.1. Writing Scripts using PowerShell

In the PowerShell world, scripting is nothing but writing a single or group of Cmdlets and combining them with various processing options. PowerShell Scripts have the .PS1 extension and can be run locally or remotely. Some examples are given below:

The below script is a basic script to extract the running processes from a system.

```
Get-WmiObject win32_process | select processname, ProcessId, CommandLine
```

In the second example, we want to see the creation date of all processes but when you add the object, it displays the date as a string. We can use the “ConvertToDateTime” method to display the date as the regular date format. We also want to sort this based on the creation date in descending order, display it in table format and all should fit to the screen properly.

```
        Get-WmiObject win32_process | select
processname,@{NAME='CreationDate';EXPRESSION={$_.ConvertToDateTime($_.CreationD
ate)}},ProcessId,CommandLine |sort CreationDate -desc | format-table -auto -
wrap
```

In yet another example, we want to save the successful logon events to a text file. In the script below, we are first defining a variable to identify where the “userprofile” directory exist and save the value for later use. The Get-WinEvent command is used to extract the event log for the specific event type, format it in a table form and save it to a text file in the user’s “desktop” folder. Time taken to complete this script – 1.75 seconds.

```
$UserDirectory = (gi env:\userprofile).value
Get-WinEvent -FilterHashtable @{Logname='security';ID=4624} | select
TimeCreated,ID,Message | ft -auto -wrap | out-file
$UserDirectory\desktop\Event-4624.txt
```

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

Windows 7 also comes with a scripting environment, Windows PowerShell Integrated Scripting Environment (ISE). ISE lets you write, test, and debug scripts (Microsoft, n.d.b).

3.2. Problems with PowerShell Scripting

Any scripting language could be used to spread malicious code, PowerShell is no exception. Due to this, Windows PowerShell by default does not allow the scripts to be run. It is controlled by what is called as an execution policy. It does provide various options to configure the system to run scripts (Microsoft, 2012).

Execution policies for computers and users can be enabled either through command line or group policy. Administrative privilege is required to change the execution policy. The execution policies are as follows:

- Restricted – This is the default policy. You can run individual commands, but not scripts.
- AllSigned – You can run scripts but the scripts must be signed by a trusted publisher.
- RemoteSigned – Scripts created on the local machine can be run. All downloaded scripts must be signed by a trusted publisher.
- Unrestricted – All types of scripts can be run.
- Undefined – In this option, policy is not set. In such cases the default execution policy of Restricted is set.

If this is too cumbersome to use, there is another option to run the PowerShell scripts. There is a not so common feature by which you are allowed to “bypass” the execution policy in PowerShell V2 through the Windows standard command interface. The command to run a script named ‘script.ps1’ is:

```
powershell.exe -ExecutionPolicy ByPass -file .\script.ps1
```

3.3. Executing scripts remotely

Windows 7 operating system provides an option to run the PowerShell scripts remotely. Microsoft uses the industry standard WS-Management Protocol to provide remote management features. This comes as a service in Windows 7, which can be

Sajeev Nair, Nair.Sajeev@Gmail.com

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

enabled either through command line or through group policy. When enabled, the machine starts a listening process over http protocol and enables the firewall to accept the connections for this process. Even though it uses http protocol for communication, the session is encrypted and authentication occur using Kerberos (Hofferle, 2012).

PowerShell remoting option could be used when the user running on the system do not have administrative rights or when you want to run the scripts on an idle system. With this option enabled, you can run the script which is stored on your local machine and have it executed on the remote machine. Processing happens on the remote machine and all outputs are collected on the local machine. The specific commands to be entered on the local machine and remote machine in order to run a script are given below:

Step	Local machine	Remote machine (Name – WRK1)
1		Enable-PSRemoting -force
2	Test-WSMan	
3	Test-WSMan –ComputerName WRK1	
4	Invoke-Command –ComputerName WRK1 –Credential domain\admin –FilePath C:\csript.ps1.	

Table 3.3.1: Commands used for PowerShell remoting.

4. Artifact collection using PowerShell

Artifacts can be collected using various methods – Windows built-in tools, PowerShell CmdLets, WMI queries and .NET classes. Wherever there are multiple methods available, it is recommended that you use multiple methods and compare results. This ensures that you have a higher possibility of identifying malicious code which tries to bypass the monitoring APIs. One good example of this is in the process and DLL queries, multiple methods include:

- Querying every DLL and asking them what process they are tied to.
- Querying every process and asking them the DLLs they have opened.
- Identifying all open files, which include DLLs for all processes.

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

Appendix A provides the various options to collect the artifacts that were discussed in section 2.1.

4.1. Sample Script

The script is written in such a way that most commands can be viewed separately and can be pasted into a PowerShell window as separate commands in case there is a need. Running separate commands enables speed and flexibility. This also ensures that you don't have to fiddle around with the defined execution policy. This is extremely important during an incident where the first responders are not trained well in the use of incident response tools and you don't want to make any system changes.

Appendix B shows the sample script and Appendix C shows the output from the sample script.

4.2. Case studies

4.2.1. Suspicious network traffic

Your perimeter monitoring picked up suspicious botnet traffic from multiple internal machines. You know the IP address that these machines are connecting to, you want to identify:

1. The machines that are making the connection to the botnet IP address.
2. The process ID, process name and port numbers that initiated this network traffic.
3. The file path from where this process was started.
4. Date and time when this process was started.
5. DLLs associated with this process with the file path.

The script that was run on the internal machines and the output are listed below. If the logged in user do not have administrative rights, the script will have to be run using PowerShell remoting method explained in section 3.3.

```
$BotNetIP = "172.20.1.21"
$CompName = (gi env:\Computername).value
$UserDirectory = (gi env:\userprofile).value
$User = (gi env:\USERNAME).value
```

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

```
$Date = (Get-Date).ToString('MM.dd.yyyy')

$head = '<style> BODY{font-family:caibri; background-color:Aliceblue;}
TABLE{border-width: 1px;border-style: solid;border-color: black;border-
collapse: collapse;} TH{font-size:1.1em; border-width: 1px;padding: 2px;border-
style: solid;border-color: black;background-color:PowderBlue} TD{border-width:
1px;padding: 2px;border-style: solid;border-color: black;background-
color:white} </style>'

$OutFile = "$UserDirectory\desktop\$CompName-$User-$Date-
NetworkConnections.html"

ConvertTo-Html -Head $head  -Title "Live Response using PowerShell" -Body "<h1>
Active Connections, Associated Processes and DLLs <p> Computer Name : $CompName
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp  User ID : $User </p>  </h1>" >
$OutFile

date | select DateTime | ConvertTo-html  -Body "<H2> Current Date and Time
</H2>" >> $OutFile


$cmd = netstat -nao | select-string $BotNetIP

 foreach ($element in $cmd)

{
       $data = $element -split ' ' | where {$_ -ne ''}


       $NetList = @{

      'Local IP : Port#'=$data[1];

      'Remote IP : Port#'=$data[2];

      'Process ID'= $data[4];

      'Process Name'=((Get-process |where {$_.ID -eq $data[4]})).Name

      'Process File Path'=((Get-process |where {$_.ID -eq $data[4]})).path

      'Process Start Time'=((Get-process |where {$_.ID -eq
      $data[4]})).starttime

      'Associated DLLs and Path'=((Get-process |where {$_.ID -eq
      $data[4]})).Modules |select @{Name='Modules';Expression={$_.filename -
      join '; ' } } |out-string

      }

      New-Object -TypeName psobject -Property $NetList |

      ConvertTo-html -Property 'Local IP : Port#', 'Remote IP : Port#','Process
      ID','Process Name','Process Start Time','Process File Path','Associated
      DLLs and File Path' -Body "<H2>     </H2>" >> $OutFile

}
date | select DateTime | ConvertTo-html  -Body "<H2> Current Date and Time
</H2>" >> $OutFile
```

Script output

Active Connections, Associated Processes and DLLs

Computer Name : Lamb-PC   User ID : lamb

Sajeev Nair, Nair.Sajeev@Gmail.com

© 2013 SANS Institute, Author retains full rights.

Author retains full rights.

Local IP : Port#	Remote IP : Port#	Process ID	Process Name	Process Start Time	Process File Path	Associated DLLs and Path
192.168.13.132:50523	172.20.1.21:80	1140	b34btbztdb0vavaw	6/11/2013 06:40:11	C:\Users\lamb\AppData\Local\Temp\ b34btbztdb0vavaw.exe	Module ------ C:\Users\lamb\AppData\Local\Temp\b34btbztdb0vavaw.exe C:\Windows\SYSTEM32\ntdll.dll C:\Windows\system32\kernel32.dll C:\Windows\system32\KERNELBASE.dll C:\Windows\system32\RPCRT4.dll C:\Windows\system32\WININET.dll C:\Windows\system32\SHLWAPI.dll C:\Windows\SYSTEM32\sechost.dll

By observing this traffic, we can identify that a malicious executable is running. It also shows the DLLs associated with the malicious code. The script completed in less than a minute.

4.2.2. Data leak

You get a specific intelligence from an employee that he noticed his colleague copying some data from his company machine to an USB drive. Since they both are working on a highly confidential merger proposal, he thinks it is related to that. Employee identified the file name as "Project-MX-proposal_V3.docx" or anything related to that.

With this knowledge, you can run the below specific commands and identify what transpired. The command outputs could be used to confirm whether this warrants a complete forensic investigation. Use PowerShell remoting feature if needed.

1. Identify if the specific file exist on the machine and the owner of the file.

```
Dir -Path C:\ -r -force -ea 0 -include *Project-MX-proposal* | select
fullname,lastwritetime,@{Name='Owner';Expression={($_ | Get-
ACL).Owner}} | sort lastwritetime -desc  | ft -auto
```

2. Collect a list of USB devices connected to the machine.

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

```
Get-ItemProperty -ea 0
hklm:\system\currentcontrolset\enum\usbstor\*\* | select
FriendlyName,PSChildName
```

3. Identify the first connected date for these devices.

```
Get-ItemProperty -ea 0
hklm:\SYSTEM\CurrentControlSet\Enum\USBSTOR\*\* | select PSChildName
| foreach-object {$P = $_.PSChildName ; Get-Content
C:\Windows\inf\setupapi.dev.log | select-string $P -SimpleMatch -
context 1 }
```

4. Identify the last connected date for these devices.

```
Get-ItemProperty -ea 0
hklm:\SYSTEM\CurrentControlSet\Enum\USBSTOR\*\* | select PSChildName
| foreach-object {$P = $_.PSChildName ;Get-WinEvent -LogName
Microsoft-Windows-DriverFrameworks-UserMode/Operational | where
{$_.message -match "$P"} | select TimeCreated, message |sort
TimeCreated -desc| ft -auto -wrap}
```

5. Identify the drive letters that were assigned to each of the USB devices.

```
Get-ItemProperty -path
hklm:\system\currentcontrolset\enum\usbstor\*\* | ForEach-Object {$P
= $_.PSChildName; Get-ItemProperty hklm:\SOFTWARE\Microsoft\"Windows
Portable Devices"\*\* |where {$_.PSChildName -like "*$P*"} | select
PSChildName,FriendlyName } | ft -auto
```

6. Find the specific user that these USB devices were connected to.

In order to do this, we have to find the Volume GUIDs for each of the mounted devices from the System\MountedDevices key. If these Volume GUIDs appear under the user's Mountpoint2 (Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2) registry location, then the drive was used by the particular user. When you have multiple users logged into the system at the same time, this key is populated for all logged in users (Fox, 2012).

7. Identify if any link files references the drive letter that the USB device used.

```
gwmi -ea 0 Win32_ShortcutFile | where {$_.FileName -like “*Project-
MX-proposal*”} | select FileName, caption, @{Name='CreationDate';
EXPRESSION={$_.ConvertToDateTime($_.CreationDate)}},@{Name=’LastAcces
sed’;EXPRESSION={$_.ConvertToDateTime($_.LastAccessed)}},@{Name=’Last
Modified’;EXPRESSION={$_.ConvertToDateTime($_.LastModified)}},Target
| sort LastModified -Descending
```

© 2013 The SANS Institute Author retains full rights.

From this command, a manual review will have to be done to identify if any drive letter matches with the drives identified in steps 5 and 6. We can also identify whether the file timestamps matches close to the device insertion time identified in steps 3 and 4.

4.2.3. Malware

One of the enterprise users reported a strange behavior while accessing a web site. The user thinks the machine downloaded a malicious code and reported to the helpdesk immediately.

In this scenario, the complete script provided in Appendix B will have to be run and the results need to be analyzed. In this situation again, use the PowerShell remoting if required. The specific findings from the analysis are given below:

Startup Applications

command	user	caption
C:\Users\lamb\AppData\Roaming\Iztugu\otez.exe	lamb-PC\lamb	{D8E86285-52AC-D466-481D-31F46A687FE2}
"C:\Program Files\Adobe\Reader 9.0\Reader\Reader_sl.exe"	Public	Adobe Reader Speed Launcher
"C:\Program Files\Common Files\Adobe\ARM\1.0\AdobeARM.exe"	Public	Adobe ARM
"C:\Program Files\Java\jre1.6.0_07\bin\jusched.exe"	Public	SunJavaUpdateSched

Prefetch Files

Name	LastAccessTime	CreationTime
SVCHOST.EXE-C871F054.pf	4/5/2013 13:15:47	4/5/2013 13:15:47
DLLHOST.EXE-40DD444D.pf	4/5/2013 13:15:57	4/5/2013 13:15:57
MORE.COM-6776F1D8.pf	6/29/2013 08:36:27	6/29/2013 08:36:27
OTEZ.EXE-8B1CFAAB.pf	7/3/2013 05:33:47	7/3/2013 05:33:47
3A2D6C8A218EBD9A178E0147629BE-	7/3/2013 05:33:48	7/3/2013

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

BD452D5D.pf		05:33:48

DNS Cache

IgnoreCase	LineNumber	Line	Pattern
True	26	Record Name . . . . . : msn.com	Record Name
True	36	Record Name . . . . . : [redacted].com	Record Name
True	46	Record Name . . . . . : www.[redacted]	Record Name

The output shows the presence of malicious code, "otez.exe" in the startup registry keys. Prefetch file listing indicates evidence of running the same malicious code. From the DNS cache, we can identify the possible web sites which may have downloaded the malicious code.

Since the analysis identified malicious code, based on the organization's policy this may warrant acquiring memory and/or a complete forensic investigation.

5. Conclusion

This paper presented various options for incident response personnel to collect artifacts that help confirm if an incident has occurred. It is fast – the sample script took only 8 minutes to run and highly scalable. With the added feature of PowerShell remoting, organizations can collect artifacts over a secure channel remotely.

PowerShell, through various Cmdlets, .NET classes and WMI objects, provides unlimited options to delve into the Windows operating system components and present the artifacts in easy to use formats. Microsoft is committed to developing PowerShell into a more robust language, which is evident from the fact that all new administrative tools for their products are built on PowerShell. PowerShell V3 has come out; it has more capabilities and more options to enumerate Windows operating system and applications (Microsoft, n.d.c).

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

More research is required in the use of PowerShell for live response, which will ultimately benefit organizations to identify threats more efficiently.

6. References

AICPA (n.d.). An executive overview of GAPP. Retrieved from http://www.aicpa.org/InterestAreas/InformationTechnology/Resources/Privacy/GenerallyAcceptedPrivacyPrinciples/DownloadableDocuments/10261378ExecOverviewGAPP.pdf

Carvey, H. (2009). Windows Forensic Analysis. Burlington, MA: Syngress

Carvey, H. (2011). Windows registry forensics: advanced digital forensic analysis of the windows registry. [Books24x7 version] Available from http://common.books24x7.com/toc.aspx?bookid=41894.

Cichonski, P., Millar, T., Grance, T., Scarfone, K. (2012). Computer Security Incident Handling Guide. Retrieved from http://csrc.nist.gov/publications/nistpubs/800-61rev2/SP800-61rev2.pdf

E-Fence. (n.d.a.). Helix3 Enterprise. Retrieved from http://www.e-fense.com/h3-enterprise.php

E-Fence. (n.d.b.). Live Response. Retrieved from http://www.e-fense.com/live-response.php

Foolmoon. (n.d.). The Windows Forensic Toolchest. Retrieved from http://www.foolmoon.net/security/wft/index.html

Fox, J. (2012). Automating Windows Registry correlation and interpretation. Retrieved from http://digitalfire.ucd.ie/?p=337

GRR. (n.d.). Retrieved from https://code.google.com/p/grr/

Guidance. (n.d.). EnCase Enterprise. Retrieved from http://www.guidancesoftware.com/encase-enterprise.htm#tab=0

Hofferle, J. (2012). An Introduction to PowerShell Remoting: Part One. Retrieved from http://blogs.technet.com/b/heyscriptingguy/archive/2012/07/23/an-introduction-to-powershell-remoting-part-one.aspx

Jones, K., Bejtlich, R., Rose, C. (2006). Real Digital Forensics: Computer Security and Incident Response. Available from:

© 2013 The SANS Institute Author retains full rights.

http://www.pearsonhighered.com/educator/product/Real-Digital-Forensics-Computer-Security-and-Incident-Response/9780321240699.page#sthash.3G1rrOkl.dpuf

Mandiant M-Trends. (2013). 2013 threat report. Retrieved from https://www.mandiant.com/resources/m-trends/

Mandiant. (n.d.). Mandiant for Intelligent Response. Retrieved from https://www.mandiant.com/products/mandiant-platform/intelligent-response

Microsoft. (n.d.a.). Scripting with Windows PowerShell. Retrieved from http://technet.microsoft.com/en-US/scriptcenter/dd742419.aspx

Microsoft. (n.d.b.). Windows PowerShell. Retrieved from http://msdn.microsoft.com/en-us/library/windows/desktop/dd835506(v=vs.85).aspx

Microsoft. (n.d.c.). Description of Windows Management Framework 3.0. Retrieved from http://support.microsoft.com/kb/2506143

Microsoft. (2012). Execution Policy. Retrieved from http://technet.microsoft.com/en-us/library/hh847748.aspx

MIR-ROR. (n.d.). Retrieved from http://mirror.codeplex.com/

Ponemon Institute. (2012). 2012 Cost of Cyber Crime Study: United States. Benchmark Study of U.S. Companies. Retrieved from http://www.ponemon.org/local/upload/file/2012_US_Cost_of_Cyber_Crime_Study_FINAL6%20.pdf

RPIER. (n.d.). Retrieved from http://code.google.com/p/rapier/

Techpathways. (n.d.). ProDiscover Incident Response. Retrieved from http://www.techpathways.com/DesktopDefault.aspx?tabindex=3&tabid=12

Trustwave Global Security Report. (2013). 2013 Global Security Report. Retrieved from http://www2.trustwave.com/rs/trustwave/images/2013-Global-Security-Report.pdf

Verizon Data Breach Report. (2013). 2013 Data breach Investigations Report. Retrieved from http://www.verizonenterprise.com/DBIR/2013/

Walters, A., Petroni, N. (2007). Volatools: Integrating volatile memory forensics into

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

the digital investigation process. Retrieved from http://www.blackhat.com/presentations/bh-dc-07/Walters/Paper/bh-dc-07-Walters-WP.pdf

© 2013 SANS Institute, Author retains full rights.

Sajeev Nair, Nair.Sajeev@Gmail.com

© 2013 The SANS Institute Author retains full rights.

7. Appendices

Appendix A: Artifact collection using PowerShell

1. Machine information and Operating system information
   The information collected should include artifacts such as machine name, OS version, licensed organization, OS install date, boot time, time zone, domain name the machine is logged into, etc. While there are multiple PowerShell Cmdlets to get this information, Windows 7 already has a built-in tool that captures all these information - systeminfo
2. User accounts and current login information
   There is a WMI class known as Win32_UserProfile, which can be queried using Get-WmiObject Cmdlet to get this information.
3. Network configuration and connectivity information
   Network configuration can be queried through another WMI class, Win32_NetworkAdapterConfiguration.
4. Anti-Virus application status and related logs
   This depends on where the log file is. If it is part of Windows application log, it can be queried through Get-WinEvent. If it is a regular text file, it can be accessed through the Get-Content Cmdlet.
5. Startup applications
   WMI class, Win32_StartupCommand captures the startup locations and the values. Additional registry locations for 64 bit operating systems, which can be queried through Get-ItemProperty are given below:
   hklm:\software\wow6432node\microsoft\windows\currentversion\run
   hklm:\software\Wow6432Node\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
   hklm:\software\wow6432node\microsoft\windows\currentversion\runonce
   hkcu:\software\wow6432node\microsoft\windows\currentversion\run
   hkcu:\software\Wow6432Node\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
   hkcu:\software\wow6432node\microsoft\windows\currentversion\runonce

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

6. Running process related information
   Multiple methods can be used to capture this information.
   - Get-Process
   - Win32_Process WMI class
   - .NET class, system.diagnostics.process
   - TASKLIST, which is a standard Windows built-in tool
7. Running services related information
   Get-Service Cmdlet or Win32_Services WMI class can be queried to get this information.
8. Drivers installed and running
   "driverquery" is an in-built Windows tool, which lists the installed drivers, the startup mode, path where it exists and date of install.
9. DLLs created
   Multiple methods can be used to capture this information.
   - Get-ChildItem Cmdlet can be used to get a listing of all DLLs that exist in the system along with their MAC timestamps.
   - TASKLIST with the M option can be used if the objective is to identify the DLLs that map to a process.
   - The WMI class, Win32_Process can also be queried to get the DLLs attached to a process.
   - .NET class, system.diagnostics.process
10. Open files
    Windows 7 has a built-in command "openfiles". It is not enabled by default; a reboot is required to take the command into effect.
11. Open shares
    WMI class, Win32_Share can be queried to get the shares open on a machine.
12. Mapped drives
    Mapped drives are stored in the below registry location. This registry entry can be queried through Get-ItemProperty Cmdlet
    hkcu:\software\Microsoft\Windows\CurrentVersion\explorer\Map Network Drive MRU

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

13. Scheduled jobs

    Win32_ScheduledJob is the WMI class that can be queried to get this information. The event log, Microsoft-Windows-TaskScheduler/ Operational also captures the scheduled tasks.

14. Active network connections and related process

    Windows standard command "netstat –nao" can be used to get the IP address, port number and the process IDs. The process ID can be further looked up against the Get-Process Cmdlet to get additional information in regards to the process.

15. Hotfixes applied

    Get-Hotfix Cmdlet retrieves this information.

16. Installed applications

    The uninstall registry key can retrieve this information.
    hklm:\software\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\

17. Link files created

    WMI class, Win32_ShortcutFile lists the link files created.

18. Packed files

    In order to identify the packed files, we have to use .NET Framework classes. The file attributes of "compressed" or "encrypted" may indicate that it is a packed file.

19. USB related

    The below registry location stores the USB devices connected to the machine.
    hklm:\system\currentcontrolset\enum\usbstor
    Operating system logs the driver installations related to the USB devices in the setupapi.dev.log file. This can be queried to understand when the device was connected to the system.

20. Shadow copies created

    WMI class, Win32_ShadowCopy lists the shadow copies created. It lists the number of shadow copies and the creation dates.

21. Prefetch files and timestamps

    Get-ChildItem can be used to list the Prefetch files. While this is not an analysis of Prefetch files, it can be used to identify the Prefetch files and the last access time.

© 2013 The SANS Institute Author retains full rights.

22. DNS cache

    Windows standard command line tool, “ipconfig /displaydns” will display the DNS cache entries.

23. List of available logs and last write times

    Logs are viewed through the Get-WinEvent Cmdlet. It can also list the logs that are updated and the size of each log.

24. Firewall configuration

    Windows netsh command, “netsh firewall” is the best option to identify the firewall configuration.

25. Audit policy

    Windows in-built command, “auditpol” lists the audit policy defined on the machine.

26. Temporary Internet files and Cookies

    Listing of files found under the temporary Internet folder can be done using the Get-ChildItem Cmdlet. The folder lists the temporary files opened through multiple applications. The same method can be used to list the Cookies folder.

27. Typed URLs

    URLs typed on the address bar are stored in the below registry key:
    hkcu:\Software\Microsoft\Internet Explorer\TypedUrls

28. Important registry keys

    There are many registry keys of interest; some of the major ones are listed below:

    - hkcu:\Software\Microsoft\Windows\CurrentVersion\Internet Settings
    - hkcu:\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\EscDomains
    - hklm:\Software\Microsoft\Windows NT\CurrentVersion\Windows
    - hklm:\Software\Microsoft\Windows\CurrentVersion\policies\system
    - hklm:\Software\Microsoft\Active Setup\Installed Components
    - hklm:\Software\Microsoft\Windows\CurrentVersion\App Paths
    - hklm:\software\microsoft\windows nt\CurrentVersion\winlogon
    - hklm:\software\microsoft\security center\svc
    - hkcu:\Software\Microsoft\Windows\CurrentVersion\Explorer\TypedPaths

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

- hkcu:\Software\Microsoft\Windows\CurrentVersion\explorer\RunMru
- hklm:\Software\Microsoft\Windows\CurrentVersion\explorer\Startmenu
- hklm:\System\CurrentControlSet\Control\Session Manager
- hklm:\Software\Microsoft\Windows\CurrentVersion\explorer\Shell Folders
- hklm:\Software\Microsoft\Windows\CurrentVersion\Shell Extensions\Approved
- hklm:\System\CurrentControlSet\Control\Session Manager\AppCertDlls
- hklm:\ Software \Classes\exefile\shell\open\command
- hklm:\BCD00000000
- hklm:\system\currentcontrolset\control\lsa
- hklm:\ Software \Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects
- hklm:\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects
- hkcu:\Software\Microsoft\Internet Explorer\Extensions
- hklm:\Software\Microsoft\Internet Explorer\Extensions
- hklm:\Software\Wow6432Node\ Microsoft\Internet Explorer\Extensions

29. File Timeline

    Get-ChildItem can be used to collect the files with a particular timestamp.

30. Important event logs

    Some of the common event logs that you want to collect as part of live response are given below:

    - Logon events
    - Logon failure events
    - Time change events
    - Application crashes
    - Process execution
    - Service control manager events
    - Windows-Application-Experience/Program-Inventory events

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

- Task scheduler events
- Terminal services events
- User creation
- Logon using explicit credentials
- Privilege use events
- DNS – failed resolution events
- WFP events

Appendix B: Sample PowerShell script

```
<#

Live Response Script Desktop

Author: Sajeev.Nair - Nair.Sajeev@gmail.com

Version : 2.0 for PowerShell V2

#>


write-host ""

Write-host "**** Script Started ****"

# Global Variables used in this script

$CompName = (gi env:\Computername).Value

$UserDirectory = (gi env:\userprofile).value

$User = (gi env:\USERNAME).value

$Date = (Get-Date).ToString('MM.dd.yyyy')

$head = '<style> BODY{font-family:caibri; background-color:Aliceblue;} 
TABLE{border-width: 1px;border-style: solid;border-color: black;border-
collapse: collapse;} TH{font-size:1.1em; border-width: 1px;padding: 2px;border-
style: solid;border-color: black;background-color:PowderBlue} TD{border-width: 
1px;padding: 2px;border-style: solid;border-color: black;background-
color:white} </style>'

$OutLevel1 = "$UserDirectory\desktop\$CompName-$User-$Date-Level1.html"

$TList = @(tasklist /V /FO CSV | ConvertFrom-Csv)

$ExecutableFiles = @("*.EXE","*.COM","*.BAT","*.BIN",
"*.JOB","*.WS",".WSF","*.PS1",".PAF","*.MSI","*.CGI","*.CMD","*.JAR","*.JSE","*
.SCR","*.SCRIPT","*.VB","*.VBE","*.VBS","*.VBSCRIPT","*.DLL")

# Setting HTML report format

ConvertTo-Html -Head $head  -Title "Live Response script for $CompName.$User" -
Body "<h1> Live Forensics Script <p> Computer Name : $CompName
```

© 2013 The SANS Institute Author retains full rights.

```
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp  User ID : $User </p>  </h1>" >
$OutLevel1


# Main Routine

# Record start time of collection

date | select DateTime | ConvertTo-html  -Body "<H2> Current Date and Time
</H2>" >> $OutLevel1

openfiles /local on

systeminfo /FO CSV | ConvertFrom-Csv | select-object * -ExcludeProperty
'Hotfix(s)','Network Card(s)' | ConvertTo-html  -Body "<H2> System Information
</H2>" >> $OutLevel1

gwmi -ea 0 Win32_UserProfile | select LocalPath, SID,@{NAME='last
used';EXPRESSION={$_.ConvertToDateTime($_.lastusetime)}} | ConvertTo-html  -
Body "<H2> User accounts and current login Information  </H2>" >> $OutLevel1

gwmi -ea 0 Win32_NetworkAdapterConfiguration |where{$_.IPEnabled -eq 'True'} |
select DHCPEnabled,@{Name='IpAddress';Expression={$_.IpAddress -join ';
'}},@{Name='DefaultIPgateway';Expression={$_.DefaultIPgateway -join ';
'}},DNSDomain  | ConvertTo-html  -Body "<H2> Network Configuration Information
</H2>" >> $OutLevel1

gwmi -ea 0 Win32_StartupCommand | select command,user,caption | ConvertTo-html
-Body "<H2> Startup Applications  </H2>" >> $OutLevel1

gp -ea 0 'hklm:\software\wow6432node\microsoft\windows\currentversion\run' |
select * -ExcludeProperty PS* | ConvertTo-html  -Body "<H2> Startup
Applications - Additional for 64 bit Systems  </H2>" >> $OutLevel1

gp -ea 0
'hklm:\software\Wow6432Node\Microsoft\Windows\CurrentVersion\Policies\Explorer\
Run' | select * -ExcludeProperty PS* | ConvertTo-html  -Body "<H2> Startup
Applications - Additional for 64 bit Systems  </H2>" >> $OutLevel1

gp -ea 0 'hklm:\software\wow6432node\microsoft\windows\currentversion\runonce'
| select * -ExcludeProperty PS* | ConvertTo-html  -Body "<H2> Startup
Applications - Additional for 64 bit Systems  </H2>" >> $OutLevel1

gp -ea 0 'hkcu:\software\wow6432node\microsoft\windows\currentversion\run' |
select * -ExcludeProperty PS* | ConvertTo-html  -Body "<H2> Startup
Applications - Additional for 64 bit Systems  </H2>" >> $OutLevel1

gp -ea 0
'hkcu:\software\Wow6432Node\Microsoft\Windows\CurrentVersion\Policies\Explorer\
Run' | select * -ExcludeProperty PS* | ConvertTo-html  -Body "<H2> Startup
Applications - Additional for 64 bit Systems  </H2>" >> $OutLevel1

gp -ea 0 'hkcu:\software\wow6432node\microsoft\windows\currentversion\runonce'
| select * -ExcludeProperty PS* | ConvertTo-html  -Body "<H2> Startup
Applications - Additional for 64 bit Systems  </H2>" >> $OutLevel1


$cmd = netstat -nao | select-string "ESTA"

 foreach ($element in $cmd)
 {

 $data = $element -split ' ' | where {$_ -ne ''}


 New-Object -TypeName psobject -Property @{

 'Local IP : Port#'=$data[1];

 'Remote IP : Port#'=$data[2];

 'Process ID'= $data[4];
```

Sajeev Nair, Nair.Sajeev@Gmail.com

© 2013 SANS Institute, Author retains full rights.

```
 'Process Name'=((Get-process |where {$_.ID -eq $data[4]})).Name

 'Process File Path'=((Get-process |where {$_.ID -eq $data[4]})).path

 'Process Start Time'=((Get-process |where {$_.ID -eq $data[4]})).starttime

 #'Process File Version'=((Get-process |where {$_.ID -eq
$data[4]})).FileVersion

 'Associated DLLs and File Path'=((Get-process |where {$_.ID -eq
$data[4]})).Modules |select @{Name='Module';Expression={$_.filename -join '; '
} } |out-string

    } | ConvertTo-html -Property 'Local IP : Port#', 'Remote IP :
Port#','Process ID','Process Name','Process Start Time','Process File
Path','Associated DLLs and File Path' -Body "<H2>     </H2>" >> $OutLevel1
}


gwmi -ea 0 win32_process | select
processname,@{NAME='CreationDate';EXPRESSION={$_.ConvertToDateTime($_.CreationD
ate)}},ProcessId,ParentProcessId,CommandLine,sessionID |sort ParentProcessId -
desc | ConvertTo-html  -Body "<H2> Running Processes sorted by ParentProcessID
</H2>" >> $OutLevel1

gwmi -ea 0 win32_process | where {$_.name -eq 'svchost.exe'} | select ProcessId
|foreach-object {$P = $_.ProcessID ;gwmi win32_service |where {$_.processId -eq
$P} | select processID,name,DisplayName,state,startmode,PathName} | ConvertTo-
html  -Body "<H2> Running SVCHOST and associated Processes  </H2>" >>
$OutLevel1

gwmi -ea 0 win32_Service  | select Name,ProcessId,State,DisplayName,PathName |
sort state | ConvertTo-html  -Body "<H2> Running Services - Sorted by State
</H2>" >> $OutLevel1

driverquery.exe /v /FO CSV | ConvertFrom-CSV | Select 'Display Name','Start
Mode', Path | sort Path | ConvertTo-html  -Body "<H2> Drivers running, Startup
mode and Path - Sorted by Path  </H2>" >> $OutLevel1

gci -r -ea 0 c:\ -include *.dll | select
Name,CreationTime,LastAccessTime,Directory | sort CreationTime -desc | select -
first 50 | ConvertTo-html  -Body "<H2> Last 50 DLLs created - Sorted by
CreationTime </H2>" >> $OutLevel1

openfiles /query > "$UserDirectory\desktop\$CompName-$User-$Date-OpenFiles.txt"

gwmi -ea 0 Win32_Share | select name,path,description | ConvertTo-html  -Body
"<H2> Open Shares  </H2>" >> $OutLevel1

gp -ea 0 'hkcu:\Software\Microsoft\Windows\CurrentVersion\explorer\Map Network
Drive MRU' | select * -ExcludeProperty PS* | ConvertTo-html  -Body "<H2> Mapped
Drives  </H2>" >> $OutLevel1

gwmi -ea 0 Win32_ScheduledJob | ConvertTo-html  -Body "<H2> Scheduled Jobs
</H2>" >> $OutLevel1

get-winevent -ea 0 -logname Microsoft-Windows-TaskScheduler/ Operational |
select TimeCreated,ID,Message | ConvertTo-html  -Body "<H2> Scheduled task
events    </H2>" >> $OutLevel1

Get-HotFix  -ea 0| Select HotfixID, Description, InstalledBy, InstalledOn |
Sort-Object InstalledOn -Descending | ConvertTo-html  -Body "<H2> HotFixes
applied - Sorted by Installed Date </H2>" >> $OutLevel1

gp -ea 0
HKLM:\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\* |
Select DisplayName,DisplayVersion,Publisher,InstallDate,InstallLocation | Sort
InstallDate -Desc  | ConvertTo-html  -Body "<H2> Installed Applications -
Sorted by Installed Date </H2>" >> $OutLevel1
```

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

```
gwmi -ea 0 Win32_ShortcutFile | select
FileName,caption,@{NAME='CreationDate';EXPRESSION={$_.ConvertToDateTime($_.Crea
tionDate)}},@{NAME='LastAccessed';EXPRESSION={$_.ConvertToDateTime($_.LastAcces
sed)}},@{NAME='LastModified';EXPRESSION={$_.ConvertToDateTime($_.LastModified)}
},Target | Where-Object  {$_.lastModified -gt ((Get-Date).addDays(-5)) }| sort
LastModified -Descending | ConvertTo-html  -Body "<H2> Link File Analysis -
Last 5 days </H2>" >> $OutLevel1

gci -Path C:\ -r -ea 0 -include $ExecutableFiles |Where {$_.Attributes -band
[IO.FileAttributes]::Compressed} | ConvertTo-html  -Body "<H2> Compressed files
</H2>" >> $OutLevel1

gci -Path C:\ -r -force -ea 0 -include $ExecutableFiles |Where {$_.Attributes -
band [IO.FileAttributes]::Encrypted} | ConvertTo-html  -Body "<H2> Encrypted
files  </H2>" >> $OutLevel1

gwmi -ea 0 Win32_ShadowCopy | select
DeviceObject,@{NAME='CreationDate';EXPRESSION={$_.ConvertToDateTime($_.InstallD
ate)}} | ConvertTo-html  -Body "<H2> ShadowCopy List </H2>" >> $OutLevel1

gci -path C:\windows\prefetch\*.pf -ea 0 | select Name,
LastAccessTime,CreationTime | sort LastAccessTime | ConvertTo-html  -Body "<H2>
Prefetch Files  </H2>" >> $OutLevel1

ipconfig /displaydns | select-string 'Record Name' | Sort | ConvertTo-html  -
Body "<H2> DNS Cache  </H2>" >> $OutLevel1

Get-WinEvent -max 50 -ea 0 -FilterHashtable @{Logname='system';ID=1014} |
select TimeCreated,ID,Message | ConvertTo-html  -Body "<H2> Event log - DNS -
failed resolution events    </H2>" >> $OutLevel1

Get-WinEvent -ea 0 -ListLog * | Where-Object {$_.IsEnabled} | Sort-Object -
Property LastWriteTime -Descending | select LogName, FileSize, LastWriteTime |
ConvertTo-html  -Body "<H2> List of available logs  </H2>" >> $OutLevel1

$la = $env:LOCALAPPDATA ;gci -r -ea 0 $la\Microsoft\Windows\'Temporary Internet
Files' | select Name, LastWriteTime, CreationTime,Directory| Where-Object
{$_.lastwritetime -gt ((Get-Date).addDays(-5)) }| Sort creationtime -Desc  |
ConvertTo-html  -Body "<H2> Temporary Internet Files - Last 5 days - Sorted by
CreationTime </H2>" >> $OutLevel1

$a = $env:APPDATA ;gci -r -ea 0 $a\Microsoft\Windows\cookies | select Name
|foreach-object {$N = $_.Name ;get-content -ea 0
$a\Microsoft\Windows\cookies\$N | select-string '/'} | ConvertTo-html  -Body
"<H2> Cookies  </H2>" >> $OutLevel1

gp -ea 0 'hkcu:\Software\Microsoft\Internet Explorer\TypedUrls' | select * -
ExcludeProperty PS* | ConvertTo-html  -Body "<H2> Typed URLs  </H2>" >>
$OutLevel1

write-host ""

Write-host "**** Script is running please wait ****"

gp -ea 0 'hkcu:\Software\Microsoft\Windows\CurrentVersion\Internet Settings' |
select * -ExcludeProperty PS* | ConvertTo-html  -Body "<H2> Important Registry
keys - Internet Settings  </H2>" >> $OutLevel1

gci -ea 0 "hkcu:SOFTWARE\Microsoft\Windows\CurrentVersion\Internet
Settings\ZoneMap\EscDomains" | select PSChildName | ConvertTo-html  -Body "<H2>
Important Registry keys - Internet Trusted Domains  </H2>" >> $OutLevel1

gp -ea 0 'hklm:\Software\Microsoft\Windows NT\CurrentVersion\Windows' | select
AppInit_DLLs | ConvertTo-html  -Body "<H2> Important Registry keys -
AppInit_DLLs  </H2>" >> $OutLevel1

gp -ea 0 hklm:\Software\Microsoft\Windows\CurrentVersion\policies\system |
select * -ExcludeProperty PS* | ConvertTo-html  -Body "<H2> Important Registry
keys - UAC Group Policy Settings </H2>" >> $OutLevel1
```

© 2013 The SANS Institute Author retains full rights.

```
gp -ea 0 'HKLM:\Software\Microsoft\Active Setup\Installed Components\*' |
select ComponentID,'(default)',StubPath  | ConvertTo-html  -Body "<H2>
Important Registry keys - Active setup Installs  </H2>" >> $OutLevel1

gp -ea 0 'hklm:\Software\Microsoft\Windows\CurrentVersion\App Paths\*' | select
PSChildName, '(default)'  | ConvertTo-html  -Body "<H2> Important Registry keys
- APP Paths keys  </H2>" >> $OutLevel1

gp -ea 0 'hklm:\software\microsoft\windows nt\CurrentVersion\winlogon\*\*' |
select '(default)',DllName  | ConvertTo-html  -Body "<H2> Important Registry
keys - DLLs loaded by Explorer.exe shell  </H2>" >> $OutLevel1

gp -ea 0 'hklm:\software\microsoft\windows nt\CurrentVersion\winlogon' | select
* -ExcludeProperty PS* | ConvertTo-html  -Body "<H2> Important Registry keys -
shell and UserInit values  </H2>" >> $OutLevel1

gp -ea 0 'hklm:\software\microsoft\security center\svc' | select * -
ExcludeProperty PS* | ConvertTo-html  -Body "<H2> Important Registry Keys -
Security center SVC values  </H2>" >> $OutLevel1

gp -ea 0 'hkcu:\Software\Microsoft\Windows\CurrentVersion\Explorer\TypedPaths'
| select * -ExcludeProperty PS* | ConvertTo-html  -Body "<H2> Important
Registry keys - Desktop Address bar history  </H2>" >> $OutLevel1

gp -ea 0 'hkcu:\Software\Microsoft\Windows\CurrentVersion\explorer\RunMru' |
select * -ExcludeProperty PS* | ConvertTo-html  -Body "<H2> Important Registry
keys - RunMRU keys  </H2>" >> $OutLevel1

gp -ea 0 'hklm:\Software\Microsoft\Windows\CurrentVersion\explorer\Startmenu' |
select * -ExcludeProperty PS* | ConvertTo-html  -Body "<H2> Important Registry
keys - Start Menu  </H2>" >> $OutLevel1

gp -ea 0 'hklm:\SYSTEM\CurrentControlSet\Control\Session Manager' | select * -
ExcludeProperty PS* | ConvertTo-html  -Body "<H2> Important Registry keys -
Programs Executed By Session Manager  </H2>" >> $OutLevel1

gp -ea 0 'hklm:\Software\Microsoft\Windows\CurrentVersion\explorer\Shell
Folders' | select * -ExcludeProperty PS* | ConvertTo-html  -Body "<H2>
Important Registry keys - Shell Folders  </H2>" >> $OutLevel1

gp -ea 0 'hkcu:\Software\Microsoft\Windows\CurrentVersion\explorer\Shell
Folders' | select startup | ConvertTo-html  -Body "<H2> Important Registry keys
- User Shell Folders 'Startup' </H2>" >> $OutLevel1

gp -ea 0 'hklm:\SOFTWARE\Microsoft\Windows\CurrentVersion\Shell
Extensions\Approved' | select * -ExcludeProperty PS* | ConvertTo-html  -Body
"<H2> Important Registry keys - Approved Shell Extentions  </H2>" >> $OutLevel1

gp -ea 0 'hklm:\System\CurrentControlSet\Control\Session Manager\AppCertDlls' |
select * -ExcludeProperty PS* | ConvertTo-html  -Body "<H2> Important Registry
keys - AppCert DLLs  </H2>" >> $OutLevel1

gp -ea 0 'hklm:\SOFTWARE\Classes\exefile\shell\open\command' | select * -
ExcludeProperty PS* | ConvertTo-html  -Body "<H2> Important Registry keys - EXE
File Shell Command Configured  </H2>" >> $OutLevel1

gp -ea 0 hklm:\SOFTWARE\Classes\HTTP\shell\open\command | select '(default)' |
ConvertTo-html  -Body "<H2> Important Registry keys - Shell Commands  </H2>" >>
$OutLevel1

gp -ea 0 hklm:\BCD00000000\*\*\*\* | select Element |select-string ‘exe’ |
select Line | ConvertTo-html  -Body "<H2> Important Registry keys - BCD Related
</H2>" >> $OutLevel1

gp -ea 0 'hklm:\system\currentcontrolset\control\lsa' | select * -
ExcludeProperty PS*| ConvertTo-html  -Body "<H2> Important Registry keys - LSA
Packages loaded  </H2>" >> $OutLevel1

gp -ea 0 'hklm:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Browser
Helper Objects\*' | select '(default)'| ConvertTo-html  -Body "<H2> Important
Registry keys - Browser Helper Objects </H2>" >> $OutLevel1
```

Sajeev Nair, Nair.Sajeev@Gmail.com

© 2013 SANS Institute, Author retains full rights.

```
gp -ea 0
'HKLM:\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Explorer\Browser
Helper Objects\*' | select '(default)' | ConvertTo-html  -Body "<H2> Important
Registry keys - Browser Helper Objects 64 Bit </H2>" >> $OutLevel1

gp -ea 0 'hkcu:\Software\Microsoft\Internet Explorer\Extensions\*' | select
ButtonText, Icon | ConvertTo-html  -Body "<H2> Important Registry keys - IE
Extensions </H2>" >> $OutLevel1

gp -ea 0 'hklm:\Software\Microsoft\Internet Explorer\Extensions\*' | select
ButtonText, Icon | ConvertTo-html  -Body "<H2> Important Registry keys - IE
Extensions </H2>" >> $OutLevel1

gp -ea 0 'hklm:\Software\Wow6432Node\Microsoft\Internet Explorer\Extensions\*'
| select ButtonText, Icon | ConvertTo-html  -Body "<H2> Important
Registry keys - IE Extensions </H2>" >> $OutLevel1

write-host ""

Write-host "**** Script is running please wait ****"

gp -ea 0 hklm:\system\currentcontrolset\enum\usbstor\*\* | select
FriendlyName,PSChildName,ContainerID | ConvertTo-html  -Body "<H2> List of USB
devices  </H2>" >> $OutLevel1

gci -Path C:\ -r -force -ea 0 -include $ExecutableFiles | Where-Object  {-not
$_.PSIsContainer -and $_.lastwritetime -gt ((Get-Date).addDays(-30)) } | select
fullname,lastwritetime,@{N='Owner';E={($_ | Get-ACL).Owner}} | sort
lastwritetime -desc | ConvertTo-html  -Body "<H2> File Timeline Executable
Files - Past 30 days  </H2>" >> $OutLevel1

gci c:\ -r -ea 0 -include $ExecutableFiles |foreach {$P = $_.fullname; get-item
$P -Stream *} |where {$_.Stream -match "Zone.Identifier"} | select filename,
stream, @{N='LastWriteTime';E={(dir $P).LastWriteTime}} | ConvertTo-html  -Body
"<H2> Downloaded executable files  </H2>" >> $OutLevel1

Get-WinEvent -max 50 -ea 0 -FilterHashtable @{Logname='security';ID=4624} |
select TimeCreated,ID,Message | ConvertTo-html  -Body "<H2> Event log - Account
logon    </H2>" >> $OutLevel1

Get-WinEvent -max 50 -ea 0 -FilterHashtable @{Logname='security';ID=4625} |
select TimeCreated,ID,Message | ConvertTo-html  -Body "<H2> Event log - An
account failed to log on    </H2>" >> $OutLevel1

Get-WinEvent -max 50 -ea 0 -FilterHashtable @{Logname='security';ID=4616} |
select TimeCreated,ID,Message | ConvertTo-html  -Body "<H2> Event log - The
system time was changed    </H2>" >> $OutLevel1

Get-WinEvent -max 50 -ea 0 -FilterHashtable @{Logname='application';ID=1002} |
select TimeCreated,ID,Message | ConvertTo-html  -Body "<H2> Event log -
Application crashes   </H2>" >> $OutLevel1

Get-WinEvent -max 50 -ea 0 -FilterHashtable @{Logname='security';ID=4688} |
select TimeCreated,ID,Message | ConvertTo-html  -Body "<H2> Event log - Process
execution    </H2>" >> $OutLevel1

Get-WinEvent -max 50 -ea 0 -FilterHashtable @{Logname='security';ID=4720} |
select TimeCreated,ID,Message | ConvertTo-html  -Body "<H2> Event log - A user
account was created    </H2>" >> $OutLevel1

Get-WinEvent -max 50 -ea 0 -FilterHashtable @{Logname='security';ID=4648} |
select TimeCreated,ID,Message | ConvertTo-html  -Body "<H2> Event log - A logon
was attempted using explicit credentials    </H2>" >> $OutLevel1

Get-WinEvent -max 50 -ea 0 -FilterHashtable @{Logname='security';ID=4672} |
select TimeCreated,ID,Message | ConvertTo-html  -Body "<H2> Event log -
Privilege use 4672   </H2>" >> $OutLevel1

Get-WinEvent -max 50 -ea 0 -FilterHashtable @{Logname='security';ID=4673} |
select TimeCreated,ID,Message | ConvertTo-html  -Body "<H2> Event log -
Privilege use 4673    </H2>" >> $OutLevel1
```

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

```
Get-WinEvent -max 50 -ea 0 -FilterHashtable @{Logname='security';ID=4674} |
select TimeCreated,ID,Message | ConvertTo-html  -Body "<H2> Event log -
Privilege use 4674    </H2>" >> $OutLevel1

Get-WinEvent -max 50 -ea 0 -FilterHashtable @{Logname='system';ID=7036} |
select TimeCreated,ID,Message | ConvertTo-html  -Body "<H2> Event log - Service
Control Manager events    </H2>" >> $OutLevel1

Get-WinEvent -max 50 -ea 0 -FilterHashtable @{Logname='system';ID=64001} |
select TimeCreated,ID,Message | ConvertTo-html  -Body "<H2> Event log - WFP
events    </H2>" >> $OutLevel1

get-winevent -ea 0 -logname Microsoft-Windows-Application-Experience/Program-
Inventory | select TimeCreated,ID,Message | ConvertTo-html  -Body "<H2>
Application inventory events    </H2>" >> $OutLevel1

get-winevent -ea 0 -logname Microsoft-Windows-TerminalServices-
LocalSessionManager | select TimeCreated,ID,Message | ConvertTo-html  -Body
"<H2> Terminal services events    </H2>" >> $OutLevel1


# Record end time of collection

date | select DateTime | ConvertTo-html  -Body "<H2> Current Date and Time
</H2>" >> $OutLevel1

# Copying network connections

netstat -naob > "$UserDirectory\desktop\$CompName-$User-$Date-
NetworkConnections.txt"

# Copying Hosts file

gc $env:windir\system32\drivers\etc\hosts > "$UserDirectory\desktop\$CompName-
$User-$Date-HostsFile.txt"

# Audit Policy

auditpol /get /category:* | select-string 'No Auditing' -notmatch >
"$UserDirectory\desktop\$CompName-$User-$Date-AuditPolicy.txt"

# Firewall Config

netsh firewall show config > "$UserDirectory\desktop\$CompName-$User-$Date-
FirewallConfig.txt"


# Popup message upon completion

(New-Object -ComObject wscript.shell).popup("Script Completed")
```

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

© 2013 SANS Institute, Author retains full rights.

Appendix C: Sample output in HTML format

Live Response Script

Computer Name : LAMB-PC   User ID : lamb

Current Date and Time

*
Wednesday, July 17, 2013 04:24:27

System Information

Host Name	OS Name	OS Version	OS Manufacturer	OS Configuration	OS Build Type	Registered Owner	System Boot Time	System Manufacturer	System Model	System Type	Processor(s)	BIOS Version	Windows Directory	System Directory	Boot Device	System Locale	Input Locale	Time Zone	Page File Location(s)	Domain	Logon Server
LAMB-PC	Microsoft Windows 7 Professional N	6.1.7601 Service Pack 1 Build 7601	Microsoft Corporation	Standalone Workstation	Multiprocessor Free	lamb	7/17/2013, 04:02:04	innotek GmbH	VirtualBox	X86-based PC	1 Processor(s) Installed.,[01]: x64 Family 6 Model 23	innotek GmbH VirtualBox, 12/1/2006	C:\Windows	C:\Windows\system32	\Device\HarddiskVolume1	en-us;English (United States)	en-us;English (United States)	(UTC+01:00) Belgrade, Bratislava, Budapest, Ljubljana,	C:\pagefile.sys	WORKGROUP	\\LAMB-PC

© 2013 The SANS Institute   Author retains full rights.

User accounts and current login Information

LocalPath	SID	last used
C:\Users\lamb	S-1-5-21-4239305696-2745980338-1987368278-1001	7/17/2013 04:24:19
C:\Windows\ServiceProfiles\NetworkService	S-1-5-20	7/17/2013 04:12:48
C:\Windows\ServiceProfiles\LocalService	S-1-5-19	7/17/2013 04:12:58
C:\Windows\system32\config\systemprofile	S-1-5-18	4/5/2013 23:03:06

Network Configuration Information

DHCPEnabled	IpAddress	DefaultIPgateway	DNSDomain
True	192.168.13.132; fe80::8b8:2386:244b:42d3	192.168.13.1	private.domain

Startup Applications

command	user	caption
%ProgramFiles%\Windows Sidebar\Sidebar.exe /autoRun	NT AUTHORITY\LOCAL SERVICE	Sidebar
"C:\Program Files\Adobe\Reader 9.0\Reader\Reader_sl.exe"	Public	Adobe Reader Speed Launcher
"C:\Program Files\Common Files\Adobe\ARM\1.0\AdobeARM.exe"	Public	Adobe ARM
"C:\Program Files\Java\jre1.6.0_07\bin\jusched.exe"	Public	SunJavaUpdateSched
C:\Windows\system32\VBoxTray.exe	Public	VBoxTray

Startup Applications - Additional for 64 bit Systems

Startup Applications - Additional for 64 bit Systems

Startup Applications - Additional for 64 bit Systems

Startup Applications - Additional for 64 bit Systems

© 2013 The SANS Institute Author retains full rights.

© 2013 SANS Institute, Author retains full rights.

Startup Applications - Additional for 64 bit Systems

Startup Applications - Additional for 64 bit Systems

Local IP : Port#	Remote IP : Port#	Process ID	Process Name	Process Start Time	Process File Path	Associated DLLs and File Path
192.168.13.132: 49295	[redacted]. 61:80	2816	iexplore	7/17/2013 04:21:43	C:\Program Files\Internet Explorer\iexplore.exe	Module ------ C:\Program Files\Internet Explorer\iexplore.exe C:\Windows\SYSTEM32\ntdll.dll C:\Windows\system32\kernel32.dll C:\Windows\system32\KERNELBASE.dll C:\Windows\system32\ADVAPI32.dll C:\Windows\system32\msvcrt.dll C:\Windows\SYSTEM32\sechost.dll C:\Windows\system32\RPCRT4.dll C:\Windows\system32\USER32.dll C:\Windows\system32\GDI32.dll C:\Windows\system32\LPK.dll C:\Windows\WinSxS\x86_microsoft.vc80.crt_1fc8b3b9a1e18e3b_8.0.50727.4940_none_d08cc06a442b34fc\MSVCP80.dll C:\Program Files\Common Files\Adobe\Acrobat\ActiveX\AcroIEHelper.dll C:\Windows\system32\SXS.DLL C:\Windows\system32\ntmarta.dll C:\Windows\system32\WLDAP32.dll C:\Windows\System32\jscript9.dll C:\Windows\system32\msimtf.dll C:\Windows\system32\windowscodecs.dll C:\Windows\System32\Dxtrans.dll C:\Windows\System32\ATL.DLL C:\Windows\system32\ddrawex.dll C:\Windows\system32\DDRAW.dll C:\Windows\system32\DCIMAN32.dll C:\Windows\system32\ImgUtil.dll C:\Windows\system32\XmlLite.dll C:\Windows\system32\MSIMG32.dll

Local IP : Port#	Remote IP : Port#	Process ID	Process Name	Process Start Time	Process File Path	Associated DLLs and File Path
192.168.13.132:49303	[redacted].55:80	2984	iexplore	7/17/2013 04:20:41	C:\Program Files\Internet Explorer\iexplore.exe	Module ------ C:\Program Files\Internet Explorer\iexplore.exe C:\Windows\SYSTEM32\ntdll.dll C:\Windows\system32\kernel32.dll C:\Windows\System32\wshtcpip.dll C:\Windows\system32\NLAapi.dll C:\Windows\system32\UxTheme.dll C:\Program Files\Common Files\Adobe\Acrobat\ActiveX\AcroIEHelperShim.dll C:\Program Files\Java\jre1.6.0_07\bin\MSVCR71.dll C:\Program

© 2013 The SANS Institute Author retains full rights.

© 2013 SANS Institute, Author retains full rights.

						Files\Java\jre1.6.0_07\bin\npjpi160_07.dll C:\Program Files\Java\jre1.6.0_07\bin\jpiexp.dll C:\Program Files\Java\jre1.6.0_07\bin\deploy.dll C:\Windows\system32\wsock32.dll C:\Windows\system32\napinsp.dll C:\Windows\system32\pnrpnsp.dll C:\Windows\System32\winrnr.dll C:\Program Files\Java\jre1.6.0_07\bin\jpishare.dll C:\PROGRA~1\Java\JRE16~1.0_0\bin\client\jvm.dll C:\PROGRA~1\Java\JRE16~1.0_0\bin\hpi.dll C:\PROGRA~1\Java\JRE16~1.0_0\bin\verify.dll C:\PROGRA~1\Java\JRE16~1.0_0\bin\java.dll C:\PROGRA~1\Java\JRE16~1.0_0\bin\zip.dll C:\Program Files\Java\jre1.6.0_07\bin\awt.dll C:\Windows\system32\WINSPOOL.DRV C:\Program C:\Windows\system32\msi.dll C:\Program Files\Java\jre1.6.0_07\bin\net.dll C:\Program Files\Java\jre1.6.0_07\bin\dcpr.dll C:\Program Files\Java\jre1.6.0_07\bin\nio.dll

Running Processes sorted by ParentProcessID

ProcessName	CreationDate	ProcessId	ParentProcessId	CommandLine	sessionID
iexplore.exe	7/17/2013 04:21:43	2816	2844	"C:\Program Files\Internet Explorer\iexplore.exe" SCODEF:2844 CREDAT:79878	1
iexplore.exe	7/17/2013 04:20:40	2844	816	"C:\Program Files\Internet Explorer\iexplore.exe"	1
b34btbztdb0vavaw.exe	7/17/2013 04:24:11	2832	752	C:\Users\lamb\AppData\Local\Temp\b34btbztdb0vavaw.exe	1
wininit.exe	7/3/2013 06:12:09	392	336	wininit.exe	0
smss.exe	7/3/2013 06:11:59	264	4	\SystemRoot\System32\smss.exe	0
System Idle Process		0	0		0

Running SVCHOST and associated Processes

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

processID	name	DisplayName	state	startmode	PathName
1976	PolicyAgent	IPsec Policy Agent	Running	Manual	C:\Windows\system32\svchost.exe -k NetworkServiceNetworkRestricted
592	WinDefend	Windows Defender	Running	Auto	C:\Windows\System32\svchost.exe -k secsvcs
1176	p2pimsvc	Peer Networking Identity Manager	Running	Manual	C:\Windows\System32\svchost.exe -k LocalServicePeerNet
1176	PNRPsvc	Peer Name Resolution Protocol	Running	Manual	C:\Windows\System32\svchost.exe -k LocalServicePeerNet

Running Services - Sorted by State

Name	ProcessId	State	DisplayName	PathName
PolicyAgent	1976	Running	IPsec Policy Agent	C:\Windows\system32\svchost.exe -k NetworkServiceNetworkRestricted
Power	612	Running	Power	C:\Windows\system32\svchost.exe -k DcomLaunch
SDRSVC	0	Stopped	Windows Backup	C:\Windows\system32\svchost.exe -k SDRSVC
RpcLocator	0	Stopped	Remote Procedure Call (RPC) Locator	C:\Windows\system32\locator.exe
dot3svc	0	Stopped	Wired AutoConfig	C:\Windows\system32\svchost.exe -k LocalSystemNetworkRestricted

Drivers running, Startup mode and Path - Sorted by Path

Display Name	Start Mode	Path
Common Log (CLFS)	Boot	C:\Windows\system32\CLFS.sys
1394 OHCI Compliant Host Controller	Manual	C:\Windows\system32\drivers\1394ohci.sys
Microsoft ACPI Driver	Boot	C:\Windows\system32\drivers\ACPI.sys

Last 50 DLLs created - Sorted by CreationTime

Name	CreationTime	LastAccessTime	Directory
wsdetect.dll	4/10/2013 10:33:04	6/10/2072 02:32:34	C:\Program Files\Java\jre1.6.0_07\bin
verify.dll	4/10/2013 10:33:04	6/10/2072 02:10:40	C:\Program Files\Java\jre1.6.0_07\bin
w2k_lsa_auth.dll	4/10/2013 10:33:04	6/10/2072 02:10:40	C:\Program Files\Java\jre1.6.0_07\bin
zip.dll	4/10/2013 10:33:04	6/10/2072 02:10:40	C:\Program Files\Java\jre1.6.0_07\bin

© 2013 The SANS Institute Author retains full rights.

unpack.dll	4/10/2013 10:33:04	6/10/2072 02:10:40	C:\Program Files\Java\jre1.6.0_07\bin

Open Shares

name	path	description
ADMIN$	C:\Windows	Remote Admin
C$	C:\	Default share
IPC$		Remote IPC
Users	C:\Users	

Mapped Drives

Scheduled Jobs

Event log – Scheduled task events

HotFixes applied - Sorted by Installed Date

HotfixID	Description	InstalledBy	InstalledOn
KB2727528	Security Update	NT AUTHORITY\SYSTEM	4/5/2013 00:00:00
KB2729094	Update	NT AUTHORITY\SYSTEM	4/5/2013 00:00:00
KB2729452	Security Update	NT AUTHORITY\SYSTEM	4/5/2013 00:00:00
KB2719857	Update	NT AUTHORITY\SYSTEM	4/5/2013 00:00:00

Installed Applications - Sorted by Installed Date

Link File Analysis - Last 5 days

FileName	caption	CreationDate	LastAccessed	LastModified	Target
lamb-	c:\users\lamb\appdata\roaming\microsoft\windows\recent\lamb-	7/17/2013	7/17/2013	7/17/2013	E:\lamb-

© 2013 The SANS Institute Author retains full rights.

© 2013 SANS Institute, Author retains full rights

07.17.2013-Level1	07.17.2013-level1.lnk	04:16:23	04:16:23	04:16:23	07.17.2013-Level1.html
host	c:\users\lamb\appdata\roaming\microsoft\windows\recent\host.lnk	7/17/2013 04:13:06	7/17/2013 04:13:06	7/17/2013 04:13:06	E:\host.txt

Compressed files

Encrypted files

ShadowCopy List

DeviceObject	CreationDate
\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1	6/13/2013 10:23:23
\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2	6/25/2013 07:10:29
\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy4	6/26/2013 13:33:40

Prefetch Files

Name	LastAccessTime	CreationTime
SVCHOST.EXE-C871F054.pf	4/5/2013 13:15:47	4/5/2013 13:15:47
DLLHOST.EXE-40DD444D.pf	4/5/2013 13:15:57	4/5/2013 13:15:57
JAVAW.EXE-3B7782B5.pf	7/17/2013 04:24:12	7/17/2013 04:24:12
REGSVR32.EXE-8461DBEE.pf	7/17/2013 04:24:12	7/17/2013 04:24:12
B34BTBZTDB0VAVAW.EXE-A991B8B7.pf	7/17/2013 04:24:21	7/17/2013 04:24:21
1390349.EXE-555A88A6.pf	7/17/2013 04:25:28	7/17/2013 04:25:28
SVCHOST.EXE-A72229FD.pf	7/17/2013 04:26:06	7/17/2013 04:26:06

DNS Cache

IgnoreCase	LineNumber	Line	Filename	Path	Pattern	Context	Matches

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

True	26	Record Name . . . . . : notepad-plus-plus.org	InputStream	InputStream	Record Name		System.Text.RegularExpressions.Match[]
True	6	Record Name . . . . . : [redacted].co.[redacted]	InputStream	InputStream	Record Name		System.Text.RegularExpressions.Match[]
True	36	Record Name . . . . . : [redacted].com	InputStream	InputStream	Record Name		System.Text.RegularExpressions.Match[]
True	46	Record Name . . . . . : www.[redacted]	InputStream	InputStream	Record Name		System.Text.RegularExpressions.Match[]

Event log – DNS – failed resolution events

TimeCreated	Id	Message
7/17/2013 04:10:22	1014	Name resolution for the name _ldap._tcp.dc._msdcs.private.domain timed out after none of the configured DNS servers responded.
7/3/2013 06:12:41	1014	Name resolution for the name isatap.private.domain timed out after none of the configured DNS servers responded.

List of available logs

LogName	FileSize	LastWriteTime
Microsoft-Windows-Windows Defender/Operational	69632	7/17/2013 04:15:27
Microsoft-Windows-ReliabilityAnalysisComponent/Operational	69632	7/17/2013 04:12:23
Microsoft-Windows-WindowsBackup/ActionCenter	69632	7/17/2013 04:10:27
Microsoft-Windows-Application-Experience/Problem-Steps-Recorder	69632	4/5/2013 13:18:49

Temporary Internet Files - Last 5 days - Sorted by CreationTime

Name	LastWriteTime	CreationTime	Directory
iesqmdata0.sqm	7/17/2013 04:17:21	7/17/2013 04:17:21	C:\Users\lamb\AppData\Local\Microsoft\Windows\Temporary Internet Files\Sqm
Sqm	7/17/2013 04:17:21	4/5/2013 14:15:21	

Cookies

Sajeev Nair, Nair.Sajeev@Gmail.com

© 2013 The SANS Institute Author retains full rights.

IgnoreCase	LineNumber	Line
True	3	[redacted].com/
True	3	[redacted].net/
True	3	www.msn.com/
True	2	[redacted].exe
True	3	downloads.sourceforge.net/

Typed URLs

url1	url2	url3	url4	url5
http://[redacted]/a.php	http://[redacted]	http://[redacted].exe	http://[redacted].exe	http://www.[redacted]/catalog/account.php

Important Registry keys - Internet Settings

IE5_UA_Backup_Flag	User Agent	Email Name	PrivDiscUiShown	EnableHttp1_1	WarnOnIntranet	MimeExclusionListForCache	AutoConfigProxy	UrlEncoding	SecureProtocols	WarnonZoneCrossing	ProxyEnable	EnableAutodial	NoNetAutodial	ProxyHttp1.1
5.0	Mozilla/4.0 (compatible; MSIE 8.0; Win32)	User@	1	1	1	multipart/mixed multipart/x-mixed-replace multipart/x-byteranges	wininet.dll	0	160	0	0	0	0	1

Important Registry keys - Internet Trusted Domains

*
microsoft.com

Sajeev Nair, Nair.Sajeev@Gmail.com

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

Important Registry keys - AppInit_DLLs

*

Important Registry keys - UAC Group Policy Settings

ConsentPromptBehaviorAdmin	ConsentPromptBehaviorUser	EnableInstallerDetection	EnableLUA	EnableSecureUIAPaths	EnableUIADesktopToggle	EnableVirtualization	PromptOnSecureDesktop	ValidateAdminCodeSignatures	dontdisplaylastusername	legalnoticecaption	legalnoticetext	scforceoption	shutdownwithoutlogon	undockwithoutlogon	FilterAdministratorToken
0	3	1	0	1	0	1	0	0	0			0	1	1	0

Important Registry keys - Active setup Installs

ComponentID	(default)	StubPath
IEACCESS	Internet Explorer	C:\Windows\System32\ie4uinit.exe -UserIconConfig
BRANDING.CAB	Browser Customizations	"C:\Windows\System32\rundll32.exe" "C:\Windows\System32\iedkcs32.dll",BrandIEActiveSetup SIGNUP
MobilePk	Offline Browsing Pack	
MailNews	Microsoft Windows	"C:\Program Files\Windows Mail\WinMail.exe" OCInstallUserConfigOE
DirectDrawEx	DirectDrawEx	
HelpCont	Internet Explorer Help	

Important Registry keys - APP Paths keys

PSChildName	(default)
AcroRd32.exe	C:\Program Files\Adobe\Reader 9.0\Reader\AcroRd32.exe
IEXPLORE.EXE	C:\Program Files\Internet Explorer\IEXPLORE.EXE
javaws.exe	C:\Program Files\Java\jre1.6.0_07\bin\javaws.exe
wireshark.exe	C:\Program Files\Wireshark\wireshark.exe
WORDPAD.EXE	"C:\Program Files\Windows NT\Accessories\WORDPAD.EXE"

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

Important Registry keys - DLLs loaded by Explorer.exe shell

(default)	DllName
Wireless Group Policy	wlgpclnt.dll
Group Policy Environment	gpprefcl.dll
Group Policy Local Users and Groups	gpprefcl.dll

Important Registry keys - shell and UserInit values

ReportBootOk	Shell	PreCreateKnownFolders	Userinit	VMApplet	AutoRestartShell	Background	CachedLogonsCount	DebugServerCommand	ForceUnlockLogon	LegalNoticeCaption	LegalNoticeText	PasswordExpiryWarning	PowerdownAfterShutdown	ShutdownWithoutLogon	WinStationsDisabled	DisableCAD	scremoveoption	ShutdownFlags
1	explorer.exe	{A520A1A4-1780-4FF6-BD18-167343C5AF16}	C:\Windows\system32\userinit.exe,	SystemPropertiesPerformance.exe /pagefile	1	0 0 0	10	no	0			5	0	0	0	1	0	39

Important Registry Keys - Security center SVC values

VistaSp1	AntiVirusOverride	AntiSpywareOverride	FirewallOverride
128920187794894432	0	0	0

Important Registry keys - Desktop Address bar history

Important Registry keys - RunMRU keys

Important Registry keys - Start Menu

Sajeev Nair, Nair.Sajeev@Gmail.com

© 2013 The SANS Institute Author retains full rights.

Type	Text	Bitmap	HelpID
group	@shell32.dll,-30464	C:\Windows\system32\shell32.dll,40	windows.hlp#51132

Important Registry keys - Programs Executed By Session Manager

CriticalSectionTimeout	GlobalFlag	HeapDeCommitFreeBlockThreshold	HeapDeCommitTotalFreeThreshold	HeapSegmentCommit	HeapSegmentReserve	ProcessorControl	ResourceTimeoutCount	BootExecute	ExcludeFromKnownDlls	ObjectDirectories	ProtectionMode	NumberOfInitialSessions	SetupExecute
2592000	16384	0	0	0	0	2	648000	System.String[]	System.String[]	System.String[]	1	2	System.String[]

Important Registry keys - Shell Folders

Common Desktop	Common Start Menu	CommonVideo	Common Pictures	Common Programs	CommonMusic	Common Administrative Tools	Common Startup	Common Documents	OEM Links	Common Templates	Common AppData
C:\Users\Public\Desktop	C:\ProgramData\Microsoft\Windows\Start Menu	C:\Users\Public\Videos	C:\Users\Public\Pictures	C:\ProgramData\Microsoft\Windows\Start Menu\Programs	C:\Users\Public\Music	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Administrative Tools	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup	C:\Users\Public\Documents	C:\ProgramData\OEM Links	C:\ProgramData\Microsoft\Windows\Templates	C:\ProgramData

Important Registry keys - User Shell Folders 'Startup'

*
C:\Users\lamb\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup

Important Registry keys - Approved Shell Extentions

Sajeev Nair, Nair.Sajeev@Gmail.com

© 2013 The SANS Institute Author retains full rights.

{00C6D95F-329C-409a-81D7-C46C66EA7F33}	{08165EA0-E946-11CF-9C87-00AA005127ED}	{F5175861-2688-11d0-9C5E-00AA00A45957}	{E6FB5E20-DE35-11CF-9C87-00AA005127ED}	{7D559C10-9FE9-11d0-93F7-00AA0059CE02}	{ABBE31D0-6DAE-11D0-BECA-00C04FD940BE}	{7FC0B86E-5FA7-11d1-BC7C-00C04FD929DB}	{23170F69-40C1-278A-1000-000100020000}
	WebCheckWebCrawler	Subscription Folder	WebCheck	Code Download Agent	Subscription Mgr	WebCheck SyncMgr Handler	7-Zip Shell Extension

Important Registry keys - AppCert DLLs

Important Registry keys - EXE File Shell Command Configured

(default)	IsolatedCommand
"%1" %*	"%1" %*

Important Registry keys - Shell Commands

*
"C:\Program Files\Internet Explorer\iexplore.exe" -nohome

Important Registry keys - BCD Related

*
@{Element=\Windows\system32\winresume.exe}
@{Element=\windows\system32\winload.exe}
@{Element=\boot\memtest.exe}

Important Registry keys - LSA Packages loaded

auditbaseobjects	auditbasedirectories	crashonauditfail	fullprivilegeauditing	Bounds	LimitBlankPasswordUse	NoLmHash	Notification Packa	Security Packa	Authentication	LsaPid	SecureBoot	ProductType	disabledomaincreds	everyoneincludesanonymous	forceguest	restrictanonymous	restrictanonymoussam

Sajeev Nair, Nair.Sajeev@Gmail.com

© 2013 The SANS Institute Author retains full rights.

							ges	ges	Packages								
0	0	0	System.Byte[]	System.Byte[]	1	1	System.String[]	System.String[]	System.String[]	496	1	16	0	0	0	0	1

Important Registry keys - Browser Helper Objects

Important Registry keys - Browser Helper Objects 64 Bit

Important Registry keys - IE Extensions

Important Registry keys - IE Extensions

Important Registry keys - IE Extensions

List of USB devices

File Timeline Executable Files - Past 30 days

FullName	LastWriteTime	Owner
C:\Users\lamb\Local Settings\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Temp\1390349.exe	7/17/2013 04:25:16	BUILTIN\Administrators
C:\Users\lamb\Local Settings\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Temp\1390349.exe	7/17/2013 04:25:16	BUILTIN\Administrators
C:\Users\lamb\Local Settings\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Temp\1390349.exe	7/17/2013 04:25:16	BUILTIN\Administrators
C:\Users\lamb\Local Settings\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Temp\1390349.exe	7/17/2013 04:25:16	BUILTIN\Administrators
C:\Documents and Settings\lamb\Local Settings\Temp\1390349.exe	7/17/2013 04:25:16	BUILTIN\Administrators
C:\Documents and Settings\lamb\Local Settings\Application Data\Application Data\Application Data\Application	7/17/2013	BUILTIN\Administrators

Sajeev Nair, Nair.Sajeev@Gmail.com

© 2013 The SANS Institute Author retains full rights.

© 2013 SANS Institute, Author retains full rights.

Data\Temp\1390349.exe	04:25:16	
C:\Documents and Settings\lamb\Local Settings\Application Data\Application Data\Application Data\Application Data\Application Data\Temp\1390349.exe	7/17/2013 04:25:16	BUILTIN\Administrators

Event log - Account logon

TimeCreated	Id	Message
7/17/2013 04:29:34	4624	An account was successfully logged on. Subject: Security ID: S-1-5-18 Account Name: LAMB-PC$ Account Domain: WORKGROUP Logon ID: 0x3e7 Logon Type: 5 New Logon: Security ID: S-1-5-18 Account Name: SYSTEM Account Domain: NT AUTHORITY Logon ID: 0x3e7 Logon GUID: {00000000-0000-0000-0000-000000000000} Process Information: Process ID: 0x1e8 Process Name: C:\Windows\System32\services.exe Network Information: Workstation Name: Source Network Address: - Source Port: - Detailed Authentication Information: Logon Process: Advapi Authentication Package: Negotiate Transited Services: - Package Name (NTLM only): - Key Length: 0 This event is generated when a logon session is created. It is generated on the computer that was accessed. The subject fields indicate the account on the local system which requested the logon. This is most commonly a service such as the Server service, or a local process such as Winlogon.exe or Services.exe. The logon type field indicates the kind of logon that occurred. The most common types are 2 (interactive) and 3 (network). The New Logon fields indicate the account for whom the new logon was created, i.e. the account that was logged on. The network fields indicate where a remote logon request originated. Workstation name is not always available and may be left blank in some cases. The authentication information fields provide detailed information about this specific logon request. - Logon GUID is a unique identifier that can be used to correlate this event with a KDC event. - Transited services indicate which intermediate services have participated in this logon request. - Package name indicates which sub-protocol was used among the NTLM protocols. - Key length indicates the length of the generated session key. This will be 0 if no session key was requested.
7/17/2013 04:29:34	4624	An account was successfully logged on. Subject: Security ID: S-1-5-18 Account Name: LAMB-PC$ Account Domain: WORKGROUP Logon ID: 0x3e7 Logon Type: 5 New Logon: Security ID: S-1-5-18 Account Name: SYSTEM Account Domain: NT AUTHORITY Logon ID: 0x3e7 Logon GUID: {00000000-0000-0000-0000-000000000000} Process Information: Process ID: 0x1e8 Process Name: C:\Windows\System32\services.exe Network Information: Workstation Name: Source Network Address: - Source Port: - Detailed Authentication Information: Logon Process: Advapi Authentication Package: Negotiate Transited Services: - Package Name (NTLM only): - Key Length: 0 This event is generated when a logon session is created. It is generated on the computer that was accessed. The subject fields indicate the account on the local system which requested the logon. This is most commonly a service such as the Server service, or a local process such as Winlogon.exe or Services.exe. The logon type field indicates the kind of logon that occurred. The most common types are 2 (interactive) and 3 (network). The New Logon fields indicate the account for whom the new logon was created, i.e. the account that was logged on. The network fields indicate where a remote logon request originated. Workstation name is not always available and may be left blank in some cases. The authentication information fields provide detailed information about this specific logon request. - Logon GUID is a unique identifier that can be used to correlate this event with a KDC event. - Transited services indicate which intermediate services have participated in this logon request. - Package name indicates which sub-protocol was used among the NTLM protocols. - Key length indicates the length of the generated session key. This will be 0 if no session key was requested.

© 2013 The SANS Institute Author retains full rights.

© 2013 SANS Institute, Author retains full rights.

Event log - An account failed to log on

Event log - The system time was changed

TimeCreated	Id	Message
7/17/2013 04:10:17	4616	The system time was changed. Subject: Security ID: S-1-5-18 Account Name: LAMB-PC$ Account Domain: WORKGROUP Logon ID: 0x3e7 Process Information: Process ID: 0x2a0 Name: C:\Windows\System32\VBoxService.exe Previous Time: 2013-07-03T04:20:00.022907600Z New Time: 2013-07-17T02:10:17.362000000Z This event is generated when the system time is changed. It is normal for the Windows Time Service, which runs with System privilege, to change the system time on a regular basis. Other system time changes may be indicative of attempts to tamper with the computer.
7/3/2013 06:15:38	4616	The system time was changed. Subject: Security ID: S-1-5-19 Account Name: LOCAL SERVICE Account Domain: NT AUTHORITY Logon ID: 0x3e5 Process Information: Process ID: 0x43c Name: C:\Windows\System32\svchost.exe Previous Time: 2013-07-03T04:15:38.589014400Z New Time: 2013-07-03T04:15:38.589000000Z This event is generated when the system time is changed. It is normal for the Windows Time Service, which runs with System privilege, to change the system time on a regular basis. Other system time changes may be indicative of attempts to tamper with the computer.

Event log – Application crashes

Event log - Process execution

Event log - A user account was created

TimeCreated	Id	Message
4/5/2013 13:15:51	4720	A user account was created. Subject: Security ID: S-1-5-18 Account Name: WIN-GV5JVE93GEV$ Account Domain: WORKGROUP Logon ID: 0x3e7 New Account: Security ID: S-1-5-21-4239305696-2745980338-1987368278-1002 Account Name: HomeGroupUser$ Account Domain: lamb-PC Attributes: SAM Account Name: HomeGroupUser$ Display Name: <value not set> User Principal Name: - Home Directory: <value not set> Home Drive: <value not set> Script Path: <value not set> Profile Path: <value not set> User Workstations: <value not set> Password Last Set: <never> Account Expires: <never> Primary Group ID: 513 Allowed To Delegate To: - Old UAC Value: 0x0 New UAC Value: 0x15 User Account Control: Account Disabled 'Password Not Required' - Enabled 'Normal Account' - Enabled User Parameters: <value not set> SID History: - Logon Hours: All Additional Information: Privileges -
4/5/2013 13:15:50	4720	A user account was created. Subject: Security ID: S-1-5-18 Account Name: WIN-GV5JVE93GEV$ Account Domain: WORKGROUP Logon ID: 0x3e7 New Account: Security ID: S-1-5-21-4239305696-2745980338-1987368278-1001 Account Name: lamb Account Domain: lamb-PC Attributes: SAM Account Name: lamb Display Name: <value not set> User Principal Name: - Home Directory: <value not set> Home Drive: <value not set> Script Path: <value not set> Profile Path: <value not set> User Workstations: <value not set>

© 2013 The SANS Institute Author retains full rights.

		Password Last Set: <never> Account Expires: <never> Primary Group ID: 513 Allowed To Delegate To: - Old UAC Value: 0x0 New UAC Value: 0x15 User Account Control: Account Disabled 'Password Not Required' - Enabled 'Normal Account' - Enabled User Parameters: <value not set> SID History: - Logon Hours: All Additional Information: Privileges -

Event log - A logon was attempted using explicit credentials

TimeCreated	Id	Message
7/3/2013 06:12:24	4648	A logon was attempted using explicit credentials. Subject: Security ID: S-1-5-18 Account Name: LAMB-PC$ Account Domain: WORKGROUP Logon ID: 0x3e7 Logon GUID: {00000000-0000-0000-0000-000000000000} Account Whose Credentials Were Used: Account Name: lamb Account Domain: lamb-PC Logon GUID: {00000000-0000-0000-0000-000000000000} Target Server: Target Server Name: localhost Additional Information: localhost Process Information: Process ID: 0x1b8 Process Name: C:\Windows\System32\winlogon.exe Network Information: Network Address: 127.0.0.1 Port: 0 This event is generated when a process attempts to log on an account by explicitly specifying that account's credentials. This most commonly occurs in batch-type configurations such as scheduled tasks, or when using the RUNAS command.
6/26/2013 13:35:30	4648	A logon was attempted using explicit credentials. Subject: Security ID: S-1-5-18 Account Name: LAMB-PC$ Account Domain: WORKGROUP Logon ID: 0x3e7 Logon GUID: {00000000-0000-0000-0000-000000000000} Account Whose Credentials Were Used: Account Name: lamb Account Domain: lamb-PC Logon GUID: {00000000-0000-0000-0000-000000000000} Target Server: Target Server Name: localhost Additional Information: localhost Process Information: Process ID: 0x1b8 Process Name: C:\Windows\System32\winlogon.exe Network Information: Network Address: 127.0.0.1 Port: 0 This event is generated when a process attempts to log on an account by explicitly specifying that account's credentials. This most commonly occurs in batch-type configurations such as scheduled tasks, or when using the RUNAS command.

Event log – Privilege use 4672

TimeCreated	Id	Message
7/17/2013 04:29:34	4672	Special privileges assigned to new logon. Subject: Security ID: S-1-5-18 Account Name: SYSTEM Account Domain: NT AUTHORITY Logon ID: 0x3e7 Privileges: SeAssignPrimaryTokenPrivilege SeTcbPrivilege SeSecurityPrivilege SeTakeOwnershipPrivilege SeLoadDriverPrivilege SeBackupPrivilege SeRestorePrivilege SeDebugPrivilege SeAuditPrivilege SeSystemEnvironmentPrivilege SeImpersonatePrivilege
7/17/2013 04:15:10	4672	Special privileges assigned to new logon. Subject: Security ID: S-1-5-18 Account Name: SYSTEM Account Domain: NT AUTHORITY Logon ID: 0x3e7 Privileges: SeAssignPrimaryTokenPrivilege SeTcbPrivilege SeSecurityPrivilege SeTakeOwnershipPrivilege SeLoadDriverPrivilege SeBackupPrivilege SeRestorePrivilege SeDebugPrivilege SeAuditPrivilege SeSystemEnvironmentPrivilege SeImpersonatePrivilege

© 2013 The SANS Institute Author retains full rights.

Event log – Privilege use 4673

Event log – Privilege use 4674

Event log – WFP events

Current Date and Time

*
Wednesday, July 17, 2013 04:32:22

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.