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.
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,
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.
3. Network configuration and connectivity information.
4. Anti-Virus application status and related logs.
5. Startup applications.
7. Running services related information.
8. Drivers installed and running.
9. DLLs created.
10. Open files.
11. Open shares.
12. Mapped drives.
15. Hotfixes applied.
16. Installed applications.
17. Link files created.
18. Packed files.
19. USB related.
20. Shadow copies created.
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.

Sajeev Nair, Nair.Sajeev@Gmail.com
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).
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).
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.
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,
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.

Sajeev Nair, Nair.Sajeev@Gmail.com
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:

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

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

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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select-Object or select</td>
<td>Primarily used to select specific properties from a Cmdlet. Example: Get-Process</td>
</tr>
<tr>
<td>Select-String</td>
<td>Similar to find or grep. This command can be used to select a specific string from the output. Example: ipconfig /displaydns</td>
</tr>
<tr>
<td>ConvertTo-Html</td>
<td>Converts the output to HTML format. This command also supports defining HEAD, TITLE and BODY options. Example: Get-Process</td>
</tr>
<tr>
<td>Format-table or ft</td>
<td>This is an output option to format the display in a table form. Example: Get-Process</td>
</tr>
<tr>
<td>Where-Object or where</td>
<td>This command is used to filter the output for specific properties. Example: Get-WmiObject Win32_NetworkAdapterConfiguration</td>
</tr>
<tr>
<td>ConvertFrom-Csv</td>
<td>This command is used to convert a CSV file for formatting within PowerShell. Example: driverquery.exe /v /FO CSV</td>
</tr>
<tr>
<td>Sort-Object or sort</td>
<td>This command sorts the properties in ascending or descending order. Example: driverquery.exe /v /FO CSV</td>
</tr>
<tr>
<td>foreach-object or foreach</td>
<td>This is the “for” loop in the PowerShell world. Example: Get-Process</td>
</tr>
<tr>
<td>Get-Date</td>
<td>This command shows the current date and time. It can also be used to add or subtract days while filtering the output. Example: Get-WinEvent -FilterHashtable @{logname='system';starttime=((Get-Date).AddDays(-1))}</td>
</tr>
</tbody>
</table>

Sajeev Nair, Nair.Sajeev@Gmail.com
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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ErrorAction or EA</td>
<td>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</td>
</tr>
<tr>
<td>Recurse or r</td>
<td>This is used to do an action recursively Ex: gci -recurse -ea 0</td>
</tr>
<tr>
<td>Path</td>
<td>Path defines the directory to be used in the Cmdlet Ex: gci -path C:\ -recurse -ea 0</td>
</tr>
<tr>
<td>Force</td>
<td>Force is used to bypass the file attribute settings of hidden and system. Ex: gci -path C:\ -recurse -force -ea 0</td>
</tr>
<tr>
<td>Include</td>
<td>This parameter is used to include a specific set of files Ex: gci -path C:\ -include *.exe –r -ea 0</td>
</tr>
</tbody>
</table>
| 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 |

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...
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={$_.ConvertToDateTime($_.CreationDate)}},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
```

Sajeev Nair, Nair.Sajeev@Gmail.com
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
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 occurs 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:

<table>
<thead>
<tr>
<th>Step</th>
<th>Local machine</th>
<th>Remote machine (Name – WRK1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Enable-PSRemoting -force</td>
</tr>
<tr>
<td>2</td>
<td>Test-WSMan</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Test-WSMan –ComputerName WRK1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Invoke-Command –ComputerName WRK1 –Credential domain\admin – FilePath C:\cscrip.ps1.</td>
<td></td>
</tr>
</tbody>
</table>

**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.
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.

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

Sajeev Nair, Nair.Sajeev@Gmail.com
$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
Computer Name : $CompName
User ID : $User</h1>"

$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
    }

}

$OutFile

Script output

Active Connections, Associated Processes and DLLs

Computer Name : Lamb-PC         User ID : lamb

Sajeev Nair, Nair.Sajeev@Gmail.com
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 a 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.

   ```powershell
   Dir -Path C:\ -r -force -ea 0 -include *Project-MX-proposal* | select fullname,lastwritetime,@{Name='Owner';Expression={$1 | Get-ACL}.Owner} | sort lastwritetime -desc | ft -auto
   ```

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

Sajeev Nair, Nair.Sajeev@Gmail.com
Get-ItemProperty -ea 0 -path hklm:\system\currentcontrolset\enum\usbstor\*\* | select FriendlyName, PSChildName

3. Identify the first connected date for these devices.

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

4. Identify the last connected date for these devices.

Get-ItemProperty -ea 0 -path hklm:\SYSTEM\CurrentControlSet\Enum\USBSTOR\*\* | select PSChildName | foreach-object {$_ = $_.PSChildName; Get-WinEvent -LogName Microsoft-Windows-DriverFrameworks-UserMode/Operational | where {$_-message -match "$_"} | 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 {$_ = $_.PSChildName; Get-ItemProperty hklm:\SOFTWARE\Microsoft\Windows Portable Devices"\*\* | where {$_.PSChildName -like "*$_*"} | 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($_.CreationDate)}}, @{Name = 'LastAccessed'; Expression = {$_.ConvertToDateTime($_.LastAccessed)}}, @{Name = 'LastModified'; Expression = {$_.ConvertToDateTime($_.LastModified)}}, Target | sort LastModified -Descending

Sajeev Nair, Nair.Sajeev@Gmail.com
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

<table>
<thead>
<tr>
<th>command</th>
<th>user</th>
<th>caption</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:\Users\lamb\AppData\Roaming\Iztugu\otez.exe</td>
<td>lamb-PC\lamb</td>
<td>{D8E86285-52AC-D466-481D-31F46A687FE2}</td>
</tr>
<tr>
<td>&quot;C:\Program Files\Adobe\Reader 9.0\Reader\Reader_sl.exe&quot;</td>
<td>Public</td>
<td>Adobe Reader Speed Launcher</td>
</tr>
<tr>
<td>&quot;C:\Program Files\Common Files\Adobe\ARM\1.0\AdobeARM.exe&quot;</td>
<td>Public</td>
<td>Adobe ARM</td>
</tr>
<tr>
<td>&quot;C:\Program Files\Java\jre1.6.0_07\bin\jusched.exe&quot;</td>
<td>Public</td>
<td>SunJavaUpdateSched</td>
</tr>
</tbody>
</table>

### Prefetch Files

<table>
<thead>
<tr>
<th>Name</th>
<th>LastAccessTime</th>
<th>CreationTime</th>
</tr>
</thead>
</table>

Sajeev Nair, Nair.Sajeev@Gmail.com
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).
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


Sajeev Nair, Nair.Sajeev@Gmail.com
Live Response Using PowerShell

Sajeev Nair, Nair.Sajeev@Gmail.com


Sajeev Nair, Nair.Sajeev@Gmail.com
the digital investigation process. Retrieved from
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
```

Sajeev Nair, Nair.Sajeev@Gmail.com
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`

Sajeev Nair, Nair.Sajeev@Gmail.com
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 -ano” 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.

Sajeev Nair, Nair.Sajeev@Gmail.com
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:
   \Software\Microsoft\Internet Explorer\TypedUrls

28. Important registry keys
   There are many registry keys of interest; some of the major ones are listed below:
   - \Software\Microsoft\Windows\CurrentVersion\Internet Settings
   - \Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\EscDomains
   - \Software\Microsoft\Windows NT\CurrentVersion\Windows
   - \Software\Microsoft\Windows\CurrentVersion\policies\system
   - \Software\Microsoft\Active Setup\Installed Components
   - \Software\Microsoft\Windows\CurrentVersion\App Paths
   - \software\microsoft\windows nt\CurrentVersion\winlogon
   - \software\microsoft\security center\svc
   - \Software\Microsoft\Windows\CurrentVersion\Explorer\TypedPaths

Sajeev Nair, Nair.Sajeev@Gmail.com
• hkcuing\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
• hkcuing\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

Sajeev Nair, Nair.Sajeev@Gmail.com
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)

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

Sajeev Nair, Nair.Sajeev@Gmail.com
Live Response Using PowerShell

Sajeev Nair, Nair.Sajeev@Gmail.com

# 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 Win32UserProfile | select LocalPath, SID @{NAME='last
used';EXPRESSION={$_.ConvertToDateTime($_.Lastusetime)}} | ConvertTo-html -Body "<H2> User accounts and current login Information </H2>
</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>
</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>
</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>
</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>
</H2>" >> $OutLevel1

gp -ea 0 'hku:software\Wow6432Node\microsoft\windows\currentversion\runonce' | select * -ExcludeProperty PS* | ConvertTo-html -Body "<H2> Startup Applications - Additional for 64 bit Systems </H2>
</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
'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


```powershell
gwmi -ea 0 win32_process | select processname.@[NAME='CreationDate']\:EXPRESSION={$_._ConvertToDateTime($_._CreationDate)}.ProcessID,ParentProcessID,CommandLine,sessionID | sort ParentProcessID -desc | ConvertTo-html -Body "<H2> Running Processes sorted by ParentProcessID </H2>" >> $OutLevel1
```

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

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

```powershell
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
```

```powershell
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
```

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

```powershell
gwmi -ea 0 win32_share | select name,path,description | ConvertTo-html -Body "<H2> Open Shares </H2>" >> $OutLevel1
```

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

```powershell
gwmi -ea 0 win32_ScheduledJob | ConvertTo-html -Body "<H2> Scheduled Jobs </H2>" >> $OutLevel1
```

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

```powershell
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
```

```powershell
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
```

Sajeev Nair, Nair.Sajeev@GMail.com
Sajeev Nair, Nair.Sajeev@Gmail.com
Live Response Using PowerShell

Sajeev Nair, Nair.Sajeev@Gmail.com

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\addons\#*' | select PSChildName, (default) | ConvertTo-html -Body "<H2> Important Registry keys - Extensions</H2>" >> $OutLevel1

gp -ea 0 'hklm:\software\microsoft\windows\currentversion\app Paths\#' | select -ExcludeProperty PS* | ConvertTo-html -Body "<H2> Important Registry keys - Installed Components</H2>" >> $OutLevel1

gp -ea 0 'hklm:\software\microsoft\windows nt\currentversion\winlogon\#' | select -ExcludeProperty PS* | ConvertTo-html -Body "<H2> Important Registry keys - BCD Related</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 'hklm:\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:\systems\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 'hklm:\software\microsoft\windows\currentversion\explorer\shellFolders' | 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 Extensions</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

Live Response Using PowerShell

$OutLevel1 = "<H2> Important Registry keys - Browser Helper Objects 64 Bit</H2>">

$OutLevel1 = "<H2> Important Registry keys - IE Extensions</H2>">

$OutLevel1 = "<H2> Important Registry keys - IE Extensions</H2>">

$OutLevel1 = "<H2> Event log – Privilege use 4673</H2>">

$OutLevel1 = "<H2> Event log – Privilege use 4672</H2>">

$OutLevel1 = "<H2> Event log – System time was changed</H2>">

$OutLevel1 = "<H2> Event log – Account logon</H2>">

$OutLevel1 = "<H2> Event log – Account failed to log on</H2>">

$OutLevel1 = "<H2> Event log – Application crashes</H2>">

$OutLevel1 = "<H2> Event log – Process execution</H2>">

$OutLevel1 = "<H2> Event log – Privilege use 4672</H2>">

$OutLevel1 = "<H2> Event log – Privilege use 4673</H2>">

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

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

Write-Host "**** Script is running please wait ****"
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")
### Appendix C: Sample output in HTML format

**Live Response Script**

**Computer Name**: LAMB-PC  
**User ID**: lamb

**Current Date and Time**

<table>
<thead>
<tr>
<th>Created</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>04:24:27</td>
</tr>
</tbody>
</table>

**System Information**

<table>
<thead>
<tr>
<th>Host Name</th>
<th>OS Name</th>
<th>OS Version</th>
<th>OS Manufacturer</th>
<th>OS Config</th>
<th>OS Build Type</th>
<th>Regis dered Owner</th>
<th>System Boot Time</th>
<th>System Manufacturer</th>
<th>System Model</th>
<th>System Type</th>
<th>BIOS Version</th>
<th>Windows Directory</th>
<th>System Directory</th>
<th>Boot Device</th>
<th>System Locale</th>
<th>Input Locale</th>
<th>Time Zone</th>
<th>Page File Location(s)</th>
<th>Domain</th>
<th>Log On Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAMB-PC</td>
<td>Microsoft Windows 7 Professional N</td>
<td>6.1.7601 Service Pack 1 Build 7601</td>
<td>Microsoft Corporation</td>
<td>Standalone Workstation</td>
<td>Multiprocessor Free</td>
<td>lamb</td>
<td>7/17/2013 04:02:04</td>
<td>innotek GmbH</td>
<td>VirtualBox</td>
<td>X86-based PC</td>
<td>1 Processor(s) Installed, [01]: x64 Family 6 Model 23</td>
<td>C:\Windows</td>
<td>\Device\HarddiskVolume1</td>
<td>English (United States)</td>
<td>English (United States)</td>
<td>(UTC +01:00) Belgrade, Bratislava, Budapest, Ljubljana, C:\pagefile.sys</td>
<td>WORKGROUP</td>
<td>LAMB-PC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
User accounts and current login Information

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

Network Configuration Information

<table>
<thead>
<tr>
<th>DHCPEnabled</th>
<th>IpAddress</th>
<th>DefaultIPgateway</th>
<th>DNSDomain</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>192.168.13.132; fe80::8b8:2386:244b:42d3</td>
<td>192.168.13.1</td>
<td>private.domain</td>
</tr>
</tbody>
</table>

Startup Applications

<table>
<thead>
<tr>
<th>command</th>
<th>user</th>
<th>caption</th>
</tr>
</thead>
<tbody>
<tr>
<td>%ProgramFiles%\Windows Sidebar\Sidebar.exe /autoRun</td>
<td>NT AUTHORITY\LOCAL SERVICE</td>
<td>Sidebar</td>
</tr>
<tr>
<td>&quot;C:\Program Files\Adobe\Reader 9.0\Reader\Reader_sl.exe&quot;</td>
<td>Public</td>
<td>Adobe Reader Speed Launcher</td>
</tr>
<tr>
<td>&quot;C:\Program Files\Common Files\Adobe\ARM\1.0\AdobeARM.exe&quot;</td>
<td>Public</td>
<td>Adobe ARM</td>
</tr>
<tr>
<td>&quot;C:\Program Files\Java\jre1.6.0_07\bin\jusched.exe&quot;</td>
<td>Public</td>
<td>SunJavaUpdateSched</td>
</tr>
<tr>
<td>C:\Windows\system32\VBoxTray.exe</td>
<td>Public</td>
<td>VBoxTray</td>
</tr>
</tbody>
</table>

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

Sajeev Nair, Nair.Sajeev@Gmail.com
### Startup Applications - Additional for 64 bit Systems

<table>
<thead>
<tr>
<th>Local IP</th>
<th>Remote IP</th>
<th>Process ID</th>
<th>Process Name</th>
<th>Process Start Time</th>
<th>Process File Path</th>
<th>Associated DLLs and File Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.13.132:49295</td>
<td>83.223.104.55:80</td>
<td>2816</td>
<td>iexplore</td>
<td>7/17/2013 04:21:43</td>
<td>C:\Program Files\Internet Explorer\iexplore.exe</td>
<td>Module ------ C:\Program Files\Internet Explorer\iexplore.exe C:\Windows\SYSTEM32\ntdll.dll C:\Windows\system32\kernel32.dll C:\Windows\system32\msvcrtd.dll C:\Windows\system32\KERNELBASE.dll C:\Windows\system32\ADVAPI32.dll C:\Windows\system32\msvct.dll C:\Windows\system32\sechost.dll C:\Windows\system32\RPCRT4.dll C:\Windows\system32\USER32.dll C:\Windows\system32\GD132.dll C:\Windows\system32\LPK.dll C:\Windows\WinSxS\x86_microsoft.vc80.crt_1fc8b3b9a1e18e3b_8.0.50727.4940_none_d08cc06442bf4cf\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</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Local IP</th>
<th>Remote IP</th>
<th>Process ID</th>
<th>Process Name</th>
<th>Process Start Time</th>
<th>Process File Path</th>
<th>Associated DLLs and File Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.13.132:49303</td>
<td>61:80</td>
<td>5580</td>
<td>iexplore</td>
<td>7/17/2013 04:20:41</td>
<td>C:\Program Files\Internet Explorer\iexplore.exe</td>
<td>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 Files\Java\jre1.6.0_07\bin\MSVCR71.dll</td>
</tr>
</tbody>
</table>
### Running Processes sorted by ParentProcessID

<table>
<thead>
<tr>
<th>ProcessName</th>
<th>CreationDate</th>
<th>ProcessId</th>
<th>ParentProcessId</th>
<th>CommandLine</th>
<th>sessionID</th>
</tr>
</thead>
<tbody>
<tr>
<td>iexplore.exe</td>
<td>7/17/2013 04:21:43</td>
<td>2816</td>
<td>2844</td>
<td>&quot;C:\Program Files\Internet Explorer\iexplore.exe&quot; SCODEF:2844 CREDAT:79878</td>
<td>1</td>
</tr>
<tr>
<td>iexplore.exe</td>
<td>7/17/2013 04:20:40</td>
<td>2844</td>
<td>816</td>
<td>&quot;C:\Program Files\Internet Explorer\iexplore.exe&quot;</td>
<td>1</td>
</tr>
<tr>
<td>b34btbzd0vavaw.exe</td>
<td>7/17/2013 04:24:11</td>
<td>2832</td>
<td>752</td>
<td>C:\Users\lamb\AppData\Local\Temp\b34btbzd0vavaw.exe</td>
<td>1</td>
</tr>
<tr>
<td>wininit.exe</td>
<td>7/3/2013 06:12:09</td>
<td>392</td>
<td>336</td>
<td>wininit.exe</td>
<td>0</td>
</tr>
<tr>
<td>smss.exe</td>
<td>7/3/2013 06:11:59</td>
<td>264</td>
<td>4</td>
<td>\SystemRoot\System32\smss.exe</td>
<td>0</td>
</tr>
<tr>
<td>System Idle Process</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

### Running SVCHOST and associated Processes

Sajeev Nair, Nair.Sajeev@Gmail.com
<table>
<thead>
<tr>
<th>processID</th>
<th>name</th>
<th>DisplayName</th>
<th>state</th>
<th>startmode</th>
<th>PathName</th>
</tr>
</thead>
<tbody>
<tr>
<td>1976</td>
<td>PolicyAgent</td>
<td>IPsec Policy Agent</td>
<td>Running</td>
<td>Manual</td>
<td>C:\Windows\system32\svchost.exe -k NetworkServiceNetworkRestricted</td>
</tr>
<tr>
<td>592</td>
<td>WinDefend</td>
<td>Windows Defender</td>
<td>Running</td>
<td>Auto</td>
<td>C:\Windows\System32\svchost.exe -k secesvc</td>
</tr>
<tr>
<td>1176</td>
<td>p2pimsvc</td>
<td>Peer Networking Identity Manager</td>
<td>Running</td>
<td>Manual</td>
<td>C:\Windows\System32\svchost.exe -k LocalServicePeerNet</td>
</tr>
<tr>
<td>1176</td>
<td>PNRPsvc</td>
<td>Peer Name Resolution Protocol</td>
<td>Running</td>
<td>Manual</td>
<td>C:\Windows\System32\svchost.exe -k LocalServicePeerNet</td>
</tr>
</tbody>
</table>

### Running Services - Sorted by State

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

### Drivers running, Startup mode and Path - Sorted by Path

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Start Mode</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Log (CLFS)</td>
<td>Boot</td>
<td>C:\Windows\system32\CLFS.sys</td>
</tr>
<tr>
<td>1394 OHCI Compliant Host Controller</td>
<td>Manual</td>
<td>C:\Windows\system32\drivers\1394ohci.sys</td>
</tr>
<tr>
<td>Microsoft ACPI Driver</td>
<td>Boot</td>
<td>C:\Windows\system32\drivers\ACPI.sys</td>
</tr>
</tbody>
</table>

### Last 50 DLLs created - Sorted by CreationTime

<table>
<thead>
<tr>
<th>Name</th>
<th>CreationTime</th>
<th>LastAccessTime</th>
<th>Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsdetect.dll</td>
<td>4/10/2013 10:33:04</td>
<td>6/10/2072 02:32:34</td>
<td>C:\Program Files\Java\jre1.6.0_07\bin</td>
</tr>
<tr>
<td>verify.dll</td>
<td>4/10/2013 10:33:04</td>
<td>6/10/2072 02:10:40</td>
<td>C:\Program Files\Java\jre1.6.0_07\bin</td>
</tr>
<tr>
<td>w2k_lsa_auth.dll</td>
<td>4/10/2013 10:33:04</td>
<td>6/10/2072 02:10:40</td>
<td>C:\Program Files\Java\jre1.6.0_07\bin</td>
</tr>
<tr>
<td>zip.dll</td>
<td>4/10/2013 10:33:04</td>
<td>6/10/2072 02:10:40</td>
<td>C:\Program Files\Java\jre1.6.0_07\bin</td>
</tr>
</tbody>
</table>
Open Shares

<table>
<thead>
<tr>
<th>name</th>
<th>path</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADMIN$</td>
<td>C:\Windows</td>
<td>Remote Admin</td>
</tr>
<tr>
<td>C$</td>
<td>C:\</td>
<td>Default share</td>
</tr>
<tr>
<td>IPC$</td>
<td></td>
<td>Remote IPC</td>
</tr>
<tr>
<td>Users</td>
<td>C:\Users</td>
<td></td>
</tr>
</tbody>
</table>

Mapped Drives

Scheduled Jobs

Event log – Scheduled task events

HotFixes applied - Sorted by Installed Date

<table>
<thead>
<tr>
<th>HotfixID</th>
<th>Description</th>
<th>InstalledBy</th>
<th>InstalledOn</th>
</tr>
</thead>
<tbody>
<tr>
<td>KB2727528</td>
<td>Security Update</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>4/5/2013 00:00:00</td>
</tr>
<tr>
<td>KB2729094</td>
<td>Update</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>4/5/2013 00:00:00</td>
</tr>
<tr>
<td>KB2729452</td>
<td>Security Update</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>4/5/2013 00:00:00</td>
</tr>
<tr>
<td>KB2719857</td>
<td>Update</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>4/5/2013 00:00:00</td>
</tr>
</tbody>
</table>

Installed Applications - Sorted by Installed Date

Link File Analysis - Last 5 days

<table>
<thead>
<tr>
<th>FileName</th>
<th>caption</th>
<th>CreationDate</th>
<th>LastAccessed</th>
<th>LastModified</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>lamb-</td>
<td></td>
<td>7/17/2013</td>
<td>7/17/2013</td>
<td>7/17/2013</td>
<td>E:\lamb-</td>
</tr>
</tbody>
</table>

Sajeev Nair, Nair.Sajeev@Gmail.com
Compressed files

Encrypted files

ShadowCopy List

<table>
<thead>
<tr>
<th>DeviceObject</th>
<th>CreationDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>\GLOBALROOT\Device\HarddiskVolumeShadowCopy1</td>
<td>6/13/2013 10:23:23</td>
</tr>
<tr>
<td>\GLOBALROOT\Device\HarddiskVolumeShadowCopy2</td>
<td>6/25/2013 07:10:29</td>
</tr>
<tr>
<td>\GLOBALROOT\Device\HarddiskVolumeShadowCopy4</td>
<td>6/26/2013 13:33:40</td>
</tr>
</tbody>
</table>

Prefetch Files

<table>
<thead>
<tr>
<th>Name</th>
<th>LastAccessTime</th>
<th>CreationTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVCHOST.EXE-A72229FD.pf</td>
<td>7/17/2013 04:26:06</td>
<td>7/17/2013 04:26:06</td>
</tr>
</tbody>
</table>

DNS Cache

<table>
<thead>
<tr>
<th>IgnoreCase</th>
<th>LineNumber</th>
<th>Line</th>
<th>Filename</th>
<th>Path</th>
<th>Pattern</th>
<th>Context</th>
<th>Matches</th>
</tr>
</thead>
</table>

Sajeev Nair, Nair.Sajeev@Gmail.com
**Event log – DNS – failed resolution events**

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

**List of available logs**

<table>
<thead>
<tr>
<th>LogName</th>
<th>FileSize</th>
<th>LastWriteTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft-Windows-ReliabilityAnalysisComponent/Operational</td>
<td>69632</td>
<td>7/17/2013 04:12:23</td>
</tr>
<tr>
<td>Microsoft-Windows-WindowsBackup/ActionCenter</td>
<td>69632</td>
<td>7/17/2013 04:10:27</td>
</tr>
</tbody>
</table>

**Temporary Internet Files - Last 5 days - Sorted by CreationTime**

<table>
<thead>
<tr>
<th>Name</th>
<th>LastWriteTime</th>
<th>CreationTime</th>
<th>Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>iesqmddata0.sqm</td>
<td>7/17/2013 04:17:21</td>
<td>7/17/2013 04:17:21</td>
<td>C:\Users\lamb\AppData\Local\Microsoft\Windows\Temporary Internet Files\Sqm</td>
</tr>
<tr>
<td>Sqm</td>
<td>7/17/2013 04:17:21</td>
<td>4/5/2013 14:15:21</td>
<td>C:\Users\lamb\AppData\Local\Microsoft\Windows\Temporary Internet Files\Sqm</td>
</tr>
</tbody>
</table>

**Cookies**
<table>
<thead>
<tr>
<th>IgnoreCase</th>
<th>LineNumber</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>3</td>
<td>[\text{obfuscated].com/]</td>
</tr>
<tr>
<td>True</td>
<td>3</td>
<td>[\text{obfuscated].net/]</td>
</tr>
<tr>
<td>True</td>
<td>3</td>
<td>\text{www.msn.com/}</td>
</tr>
<tr>
<td>True</td>
<td>2</td>
<td>[\text{obfuscated].exe]</td>
</tr>
<tr>
<td>True</td>
<td>3</td>
<td>\text{downloads.sourceforge.net/}</td>
</tr>
</tbody>
</table>

**Typed URLs**

<table>
<thead>
<tr>
<th>url1</th>
<th>url2</th>
<th>url3</th>
<th>url4</th>
<th>url5</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{http://[\text{obfuscated]}pa}</td>
<td>\text{http://[\text{obfuscated]}pe}</td>
<td>\text{http://[\text{obfuscated]}e}</td>
<td>\text{http://[\text{obfuscated]}e}</td>
<td>\text{<a href="http://www.%5C%5B%5Ctext%7Bobfuscated%5D%7Dp%7D">http://www.\[\text{obfuscated]}p}</a></td>
</tr>
</tbody>
</table>

**Important Registry keys - Internet Settings**

<table>
<thead>
<tr>
<th>IE5_UA_Ba\n\nup_Fl\ng</th>
<th>User Agent</th>
<th>Email Name</th>
<th>PrivDisc UiShown</th>
<th>Enable Http1_1</th>
<th>WarnOn Intranet</th>
<th>MimeExclusion ListForCache</th>
<th>AutoConf igProxy</th>
<th>UrlEn coding</th>
<th>SecureProtocols</th>
<th>WarnonZoneCrossing</th>
<th>Proxy Enable</th>
<th>EnableAuto\ndial</th>
<th>NoNetAuto\ndial</th>
<th>Proxy Http1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.0</td>
<td>Mozilla/4.0 (compatible; MSIE 8.0; Win32)</td>
<td>User@1</td>
<td>1</td>
<td>1</td>
<td>multipart/mixed</td>
<td>multipart/x- \n mixed-replace</td>
<td>multipart/x- \nbyteranges</td>
<td>wininet.dl</td>
<td>0</td>
<td>160</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Important Registry keys - Internet Trusted Domains**

* microsoft.com
**Important Registry keys - AppInit_DLLs**

- **ConsentPromptBehaviorAdmin**: 0
- **ConsentPromptBehaviorUser**: 3
- **EnableIn-stallerDetection**: 1
- **EnableUA**: 0
- **EnableSecureUIAPaths**: 1
- **EnableUIADesktopToggle**: 0
- **EnableVirtualization**: 1
- **PromptOnSecureDesktop**: 0
- **ValidateAdminCodeSignatures**: 0
- **dontdisplaylastusename**: 1
- **legalnoticecaption**: 1
- **legalnoticetext**: 1
- **scforceoption**: 0
- **shutdownwithoutlogon**: 1
- **unlockwithoutlogon**: 0
- **FilterAdministratortrToken**: 0

---

**Important Registry keys - UAC Group Policy Settings**

<table>
<thead>
<tr>
<th>ComponentID</th>
<th>(default)</th>
<th>StubPath</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEACCESS</td>
<td>Internet Explorer</td>
<td>C:\Windows\System32\ie4uinit.exe -UserIconConfig</td>
</tr>
<tr>
<td>BRANDING.CAB</td>
<td>Browser Customizations</td>
<td>&quot;C:\Windows\System32\rundll32.exe&quot; &quot;C:\Windows\System32\iedkcs32.dll&quot;,BrandIEActiveSetup SIGNUP</td>
</tr>
<tr>
<td>MobilePk</td>
<td>Offline Browsing Pack</td>
<td></td>
</tr>
<tr>
<td>MailNews</td>
<td>Microsoft Windows</td>
<td>&quot;C:\Program Files\Windows Mail\WinMail.exe&quot; OCInstallUserConfigOE</td>
</tr>
<tr>
<td>DirectDrawEx</td>
<td>DirectDrawEx</td>
<td></td>
</tr>
<tr>
<td>HelpCont</td>
<td>Internet Explorer Help</td>
<td></td>
</tr>
</tbody>
</table>

---

**Important Registry keys - Active setup Installs**

<table>
<thead>
<tr>
<th>ComponentID</th>
<th>(default)</th>
<th>StubPath</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcroRd32.exe</td>
<td>C:\Program Files\Adobe\Reader 9.0\Reader\AcroRd32.exe</td>
<td></td>
</tr>
<tr>
<td>IEXPLORE.EXE</td>
<td>C:\Program Files\Internet Explorer\IEXPLORE.EXE</td>
<td></td>
</tr>
<tr>
<td>javaws.exe</td>
<td>C:\Program Files\Java\jre1.6.0_07\bin\javaws.exe</td>
<td></td>
</tr>
<tr>
<td>wireshark.exe</td>
<td>C:\Program Files\Wireshark\wireshark.exe</td>
<td></td>
</tr>
<tr>
<td>WORDPAD.EXE</td>
<td>&quot;C:\Program Files\Windows NT\Accessories\WORDPAD.EXE&quot;</td>
<td></td>
</tr>
</tbody>
</table>

---

**Important Registry keys - APP Paths keys**

<table>
<thead>
<tr>
<th>PSChildName</th>
<th>(default)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcroRd32.exe</td>
<td>C:\Program Files\Adobe\Reader 9.0\Reader\AcroRd32.exe</td>
</tr>
<tr>
<td>IEXPLORE.EXE</td>
<td>C:\Program Files\Internet Explorer\IEXPLORE.EXE</td>
</tr>
<tr>
<td>javaws.exe</td>
<td>C:\Program Files\Java\jre1.6.0_07\bin\javaws.exe</td>
</tr>
<tr>
<td>wireshark.exe</td>
<td>C:\Program Files\Wireshark\wireshark.exe</td>
</tr>
<tr>
<td>WORDPAD.EXE</td>
<td>&quot;C:\Program Files\Windows NT\Accessories\WORDPAD.EXE&quot;</td>
</tr>
</tbody>
</table>
Important Registry keys - DLLs loaded by Explorer.exe shell

<table>
<thead>
<tr>
<th>(default)</th>
<th>DllName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wireless Group Policy</td>
<td>wlgpclnt.dll</td>
</tr>
<tr>
<td>Group Policy Environment</td>
<td>gpprefcl.dll</td>
</tr>
<tr>
<td>Group Policy Local Users and Groups</td>
<td>gpprefcl.dll</td>
</tr>
</tbody>
</table>

Important Registry keys - shell and UserInit values

<table>
<thead>
<tr>
<th>ReportBootOk</th>
<th>Shell</th>
<th>PreCreateKnownFolders</th>
<th>Userinit</th>
<th>VMApartment</th>
<th>AutoRestartShell</th>
<th>Background</th>
<th>CacheLogosCount</th>
<th>DebugServerCommand</th>
<th>ForceUnloclLogo n</th>
<th>LegalNoticeCaption</th>
<th>LegalNoticeText</th>
<th>PasswordExpri dyWarning</th>
<th>PowerdownAfterShutdown</th>
<th>ShutdownWithoutLogon</th>
<th>WinStationsDisabled</th>
<th>DisableCAD</th>
<th>ScreenshotOption</th>
<th>ShutdownFlags</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>explorer.exe</td>
<td>{A520A1A4-1780-4FF6-BD18-167343C5AF16}</td>
<td>userinit.exe, SystemPropertiesPerformance.exe\pagefile</td>
<td>1</td>
<td>0 0 0</td>
<td>10</td>
<td>no</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>39</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Important Registry Keys - Security center SVC values

<table>
<thead>
<tr>
<th>VistaSp1</th>
<th>AntiVirusOverride</th>
<th>AntiSpywareOverride</th>
<th>FirewallOverride</th>
</tr>
</thead>
<tbody>
<tr>
<td>128920187794894432</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Important Registry keys - Desktop Address bar history

Important Registry keys - RunMRU keys

Important Registry keys - Start Menu

Sajeev Nair, Nair.Sajeev@Gmail.com
Important Registry keys - Programs Executed By Session Manager

<table>
<thead>
<tr>
<th>CriticalSectionTimeOut</th>
<th>GlobalFlag</th>
<th>HeapDeCommitFreeBlockThreshold</th>
<th>HeapDeCommitTotalFreeThreshold</th>
<th>HeapSegmentCommit</th>
<th>HeapSegmentReserve</th>
<th>ProcessOrControl</th>
<th>ResourceTimeoutCount</th>
<th>BootExecute</th>
<th>ExcludeFromKnownDlls</th>
<th>ObjectDirectories</th>
<th>ProtectionMode</th>
<th>NumberOfInitialSessions</th>
<th>SetupExecute</th>
</tr>
</thead>
<tbody>
<tr>
<td>2592000</td>
<td>16384</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>648000</td>
<td></td>
<td>System.String[]</td>
<td>System.String[]</td>
<td>System.String[]</td>
<td>1</td>
<td>2</td>
<td>System.String[]</td>
</tr>
</tbody>
</table>

Important Registry keys - Shell Folders

<table>
<thead>
<tr>
<th>CommonDesktop</th>
<th>CommonStartMenu</th>
<th>CommonVideo</th>
<th>CommonPictures</th>
<th>CommonPrograms</th>
<th>CommonMusic</th>
<th>CommonAdministrativeTools</th>
<th>CommonStartup</th>
<th>CommonDocuments</th>
<th>OEMLinks</th>
<th>CommonTemplates</th>
<th>CommonAppData</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:\Users\Public\Desktop</td>
<td>C:\ProgramData\Microsoft\Windows\Start Menu</td>
<td>C:\Users\Public\Videos</td>
<td>C:\Users\Public\Pictures</td>
<td>C:\Users\Public\Programs</td>
<td>C:\Users\Public\Music</td>
<td>C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Administrative Tools</td>
<td>C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup</td>
<td>C:\Users\Public\Documents</td>
<td>C:\ProgramData\OEMLinks</td>
<td>C:\ProgramData\Templates</td>
<td>C:\ProgramData</td>
</tr>
</tbody>
</table>

Important Registry keys - User Shell Folders 'Startup'

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

Important Registry keys - Approved Shell Extensions

Sajeev Nair, Nair.Sajeev@Gmail.com
### Important Registry keys - AppCert DLLs

### Important Registry keys - EXE File Shell Command Configured

<table>
<thead>
<tr>
<th>(default)</th>
<th>IsolatedCommand</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;%1&quot; %* &quot;%1&quot; %*</td>
<td></td>
</tr>
</tbody>
</table>

### Important Registry keys - Shell Commands

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

### Important Registry keys - BCD Related

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

### Important Registry keys - LSA Packages loaded

<table>
<thead>
<tr>
<th>auditbaseobjects</th>
<th>auditbaseobjects</th>
<th>crashtail</th>
<th>fullprivilegedauditing</th>
<th>Bounded</th>
<th>LimitBlanksPasswordUse</th>
<th>NoLoginHash</th>
<th>NotificationPacket</th>
<th>SecurityPacka</th>
<th>Authentication</th>
<th>TrustedBoots</th>
<th>ProductType</th>
<th>disabledomaincreds</th>
<th>everyoneincludesanonymous</th>
<th>forceguest</th>
<th>restrictdomainss</th>
<th>restrictanonymous</th>
</tr>
</thead>
</table>

Sajeev Nair, Nair.Sajeev@Gmail.com
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

<table>
<thead>
<tr>
<th>FullName</th>
<th>LastWriteTime</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:\Users\lamb\Local Settings\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Temp\1390349.exe</td>
<td>7/17/2013 04:25:16</td>
<td>BUILTIN\Administrators</td>
</tr>
<tr>
<td>C:\Users\lamb\Local Settings\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Temp\1390349.exe</td>
<td>7/17/2013 04:25:16</td>
<td>BUILTIN\Administrators</td>
</tr>
<tr>
<td>C:\Users\lamb\Local Settings\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Temp\1390349.exe</td>
<td>7/17/2013 04:25:16</td>
<td>BUILTIN\Administrators</td>
</tr>
<tr>
<td>C:\Users\lamb\Local Settings\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Temp\1390349.exe</td>
<td>7/17/2013 04:25:16</td>
<td>BUILTIN\Administrators</td>
</tr>
<tr>
<td>C:\Documents and Settings\lamb\Local Settings\Temp\1390349.exe</td>
<td>7/17/2013 04:25:16</td>
<td>BUILTIN\Administrators</td>
</tr>
<tr>
<td>C:\Documents and Settings\lamb\Local Settings\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Application Data\Temp\1390349.exe</td>
<td>7/17/2013 04:25:16</td>
<td>BUILTIN\Administrators</td>
</tr>
</tbody>
</table>
Event log - Account logon

<table>
<thead>
<tr>
<th>TimeCreated</th>
<th>Id</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>7/17/2013</td>
<td>4624</td>
<td>An account was successfully logged on. Subject: Security ID: S-1-5-18 Account Name: LAMB-PCS 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.</td>
</tr>
<tr>
<td>7/17/2013</td>
<td>4624</td>
<td>An account was successfully logged on. Subject: Security ID: S-1-5-18 Account Name: LAMB-PCS 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.</td>
</tr>
</tbody>
</table>
Event log - An account failed to log on

Event log - The system time was changed

<table>
<thead>
<tr>
<th>TimeCreated</th>
<th>Id</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>7/17/2013 04:10:17</td>
<td>4616</td>
<td>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.</td>
</tr>
<tr>
<td>7/3/2013 06:15:38</td>
<td>4616</td>
<td>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.</td>
</tr>
</tbody>
</table>

Event log – Application crashes

Event log - Process execution

Event log - A user account was created

<table>
<thead>
<tr>
<th>TimeCreated</th>
<th>Id</th>
<th>Message</th>
</tr>
</thead>
</table>
Event log - A logon was attempted using explicit credentials

<table>
<thead>
<tr>
<th>TimeCreated</th>
<th>Id</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>7/3/2013 06:12:24</td>
<td>4648</td>
<td>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}</td>
</tr>
<tr>
<td>6/26/2013 13:35:30</td>
<td>4648</td>
<td>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}</td>
</tr>
</tbody>
</table>

Event log – Privilege use 4672

<table>
<thead>
<tr>
<th>TimeCreated</th>
<th>Id</th>
<th>Message</th>
</tr>
</thead>
</table>
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
<table>
<thead>
<tr>
<th>Event</th>
<th>Location</th>
<th>Dates</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS Las Vegas Summer 2020</td>
<td>Las Vegas, NVUS</td>
<td>Jun 08, 2020 - Jun 13, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS New Orleans 2020</td>
<td>Online, LAUS</td>
<td>Jun 08, 2020 - Jun 13, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS OnDemand</td>
<td>Books &amp; MP3s Only, US</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
</tbody>
</table>