Kansa: Service related collectors and analysis

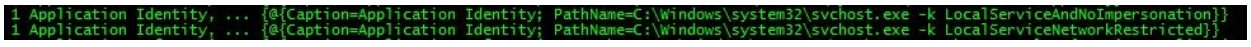
By davehull

Published: 2014-05-03 · Archived: 2026-04-06 01:00:18 UTC

In my previous post on Kansa's Autoruns collectors and analysis scripts, I mentioned that the Get-Aurounsc.ps1 collector relies on Sysinternals' Autorunsc.exe to collect data on all of the Autostart Extension Points (ASEPs) that it has catalogued. Autorunsc and its GUI sibling, Autoruns, are great tools, but they are not comprehensive, there are other ASEPs that they don't catch, so Kansa includes a few additional modules that aim to collect additional ASEPs and additional data about ASEPs.

Get-SvcAll.ps1
Runs Get-WMIObject win32_service to collect details about all services. Output is saved as XML. Some of this same data is collected by Get-Autorunsc.ps1 above, however, this will pull additional properties for each service with some of them being specific to the type of service. If a service is running, you'll get its **process id** and the **context it runs under (Local System, Local Service, etc.)**. There's even an **InstallDate** property, which is awesome, however, in my experience, it's never populated, which sucks.

For analysis of the data collected by Get-SvcAll.ps1, there are two very basic frequency analysis or stacking scripts as of this writing. They are Get-SvcAllStack.ps1 and Get-SvcStartNameStack.ps1. The former does its frequency analysis based on Service "**Captions**" and **Pathnames**. The Captions are the short friendly names you see when you look at the Services running on your system while the **Pathnames include the path to the binary and any arguments**. Here's an example from two systems where the Application Identity service has two different sets of command line arguments:



Click for larger image

Stacking by these properties across many hosts shows investigators services that may have the same Caption, but different binaries and arguments. This same kind of analysis is available in the Autoruns stacking scripts with the added benefit of stacking by file hash (e.g. MD5).

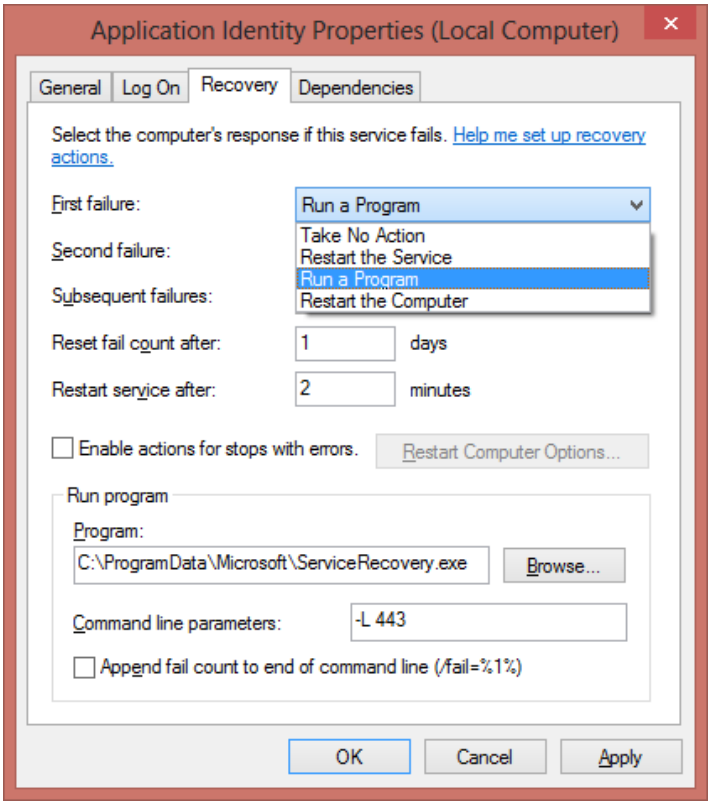
Get-SvcStartNameStack.ps1 stacks by Caption and **StartName**, the latter of which turns out to be the name of the account the service runs under.

Another Service analysis script, but not a stacker, is Get-SvcAllRunningAuto.ps1, which pulls the list of Services that were in a running state or set to start automatically when the Get-SvcAll.ps1 collector ran on the targets.

ASEPs not collected by Autorunsc:

As I mentioned above, Sysinternals' Autoruns and Autorunsc executables collect all the ASEPs they know to collect, but that is not the universe of ASEPs.

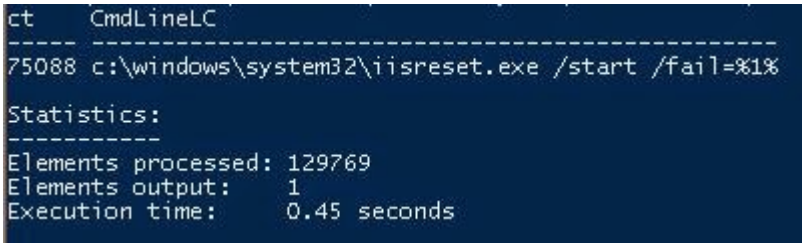
Windows Services can be configured to recover from failures. In my experience, restarting the service is the most common recovery option, but one option that adversaries can use is the "Run a Program" option as shown below:



Click for larger image

In the screen shot above, the Application Identity service is configured with a failure recovery response that will run a program called ServiceRecovery.exe from C:\ProgramData\Microsoft\ with the command line argument -L 443. This is a persistence mechanism that Autorunsc won't capture.

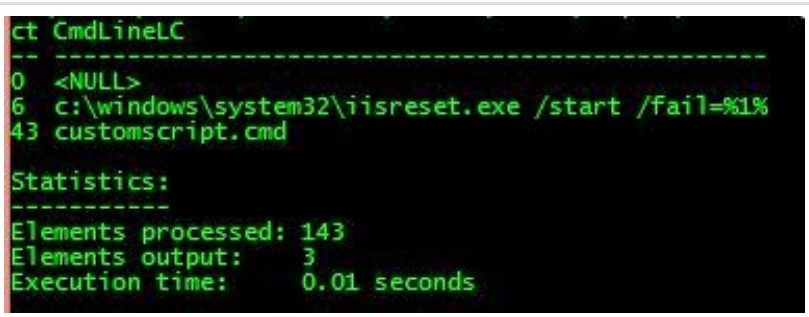
Kansa's **Get-SvcFail.ps1** collector will collect service failure recovery information from all services. Kansa includes a few analysis scripts that will stack the service failure recovery data, but the most useful one is **Get-SvcFailCmdLine.ps1**, which returns the frequency count of the program and command line parameters from all the collected service failure information. The image below shows this data from a few thousand systems:



Click for larger image

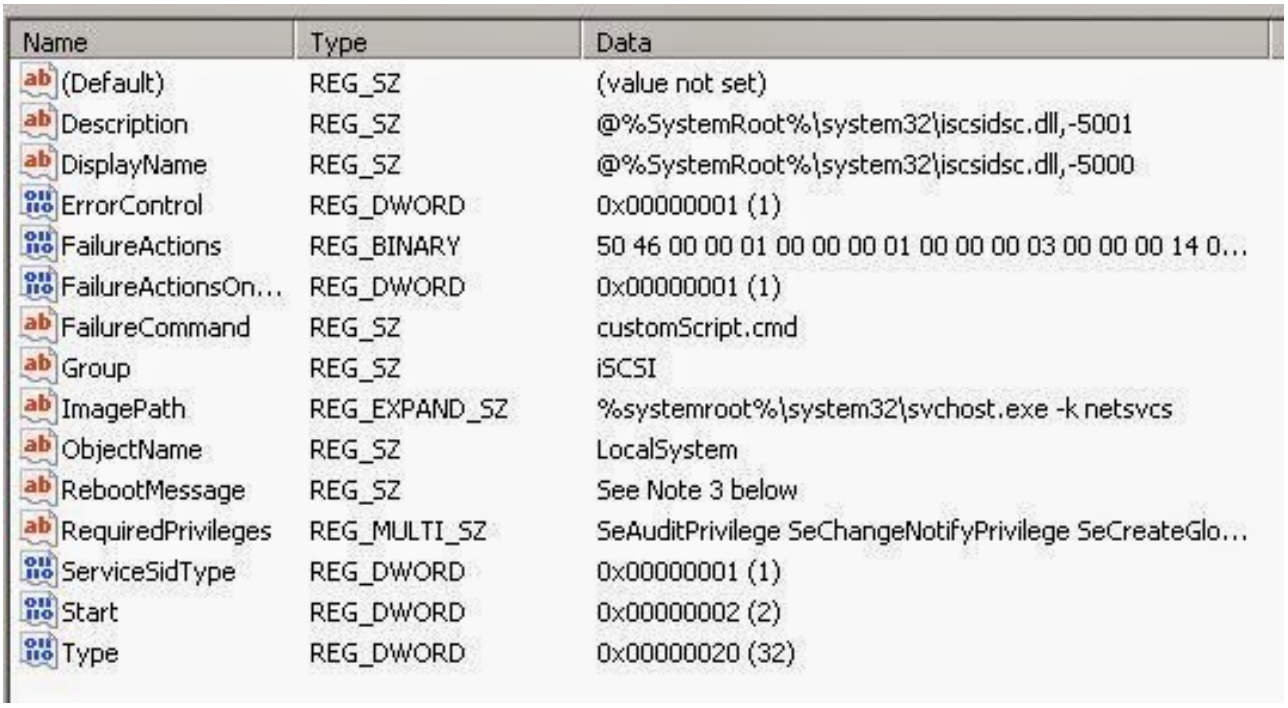
In the example there are 129769 Service Failure entries, 75088 of them have the same program and command line arguments configured as a recovery option. Seems unlikely this is malicious.

In another smaller data set, the following data was returned:



Click for larger image

I include this screen shot because I've run into the customscript.cmd entry in multiple data sets and in all the cases I've investigated, I've not yet found a service that referenced customscript.cmd anywhere in the Services GUI, but you will see services reference it in the data of their Registry key values, like the following:



Name	Type	Data
(Default)	REG_SZ	(value not set)
Description	REG_SZ	@%SystemRoot%\system32\iscsidsc.dll,-5001
DisplayName	REG_SZ	@%SystemRoot%\system32\iscsidsc.dll,-5000
ErrorControl	REG_DWORD	0x00000001 (1)
FailureActions	REG_BINARY	50 46 00 00 01 00 00 00 01 00 00 00 03 00 00 00 14 0...
FailureActionsOn...	REG_DWORD	0x00000001 (1)
FailureCommand	REG_SZ	customScript.cmd
Group	REG_SZ	iSCSI
ImagePath	REG_EXPAND_SZ	%systemroot%\system32\svchost.exe -k netsvcs
ObjectName	REG_SZ	LocalSystem
RebootMessage	REG_SZ	See Note 3 below
RequiredPrivileges	REG_MULTI_SZ	SeAuditPrivilege SeChangeNotifyPrivilege SeCreateGlo...
ServiceSidType	REG_DWORD	0x00000001 (1)
Start	REG_DWORD	0x00000002 (2)
Type	REG_DWORD	0x00000020 (32)

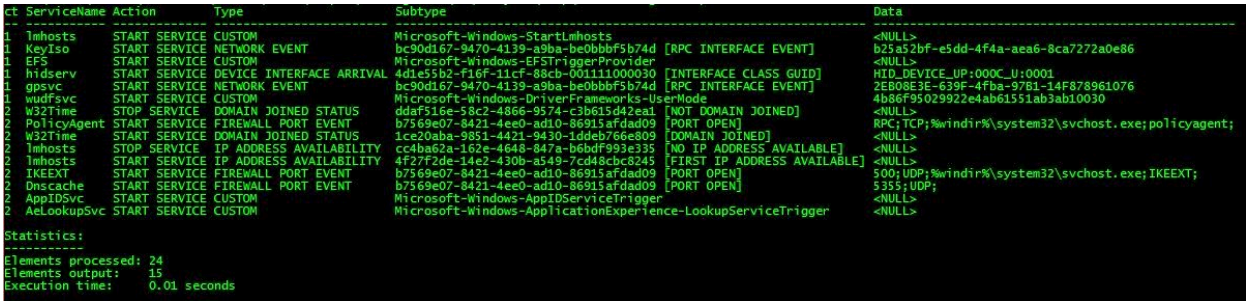
I've also searched file systems on hosts where I've seen this, but I've not found a file on disk called customScript.cmd. I wanted to mention it here in case you run across it. If you do see a reference to customscript.cmd that includes a path, you may have an adversary attempting to blend in with a common value.

The last Service related collector in Kansa, as of this writing, is **Get-SvcTrigs.ps1**, which collects another set of ASEPs that Autoruns does not collect, yet -- Service Triggers. Service Triggers are new with Windows 7 and later versions of Windows. They allow Windows Services to have more startup flexibility than the old Manual and Automatic startup modes. Now services can respond to the presence of specific hardware, group policy changes, networking events, etc. More information about Service Triggers can be found at the following links:

- http://msdn.microsoft.com/en-us/library/windows/desktop/dd405513(v=vs.85).aspx,

- http://blogs.windows.com/windows/archive/b/developers/archive/2009/10/26/windows7-trigger-start-services-part-1-introduction.aspx
- http://blogs.windows.com/windows/archive/b/developers/archive/2009/10/27/windows7-trigger-start-services-part-2-building-a-trigger-start-optimized-service.aspx.

Kansa includes a basic stacker for Service Triggers. Interpreting the data to determine what's normal and what's suspicious can be daunting and tedious. Searching on GUIDs can be of some help. Below is a frequency listing of Service Triggers from a relatively small sample, two systems.



```
ct ServiceName Action        Type                    Subtype                                                                 Data
-- ----------- ------------- ----------------------- ----------------------------------------------------------------------- ----------------------------------------------------
1  lmhosts     START SERVICE CUSTOM                  Microsoft-Windows-StartLmhosts                                          <NULL>
1  KeyIso      START SERVICE NETWORK EVENT           bc90d167-9470-4139-a9ba-be0bbbf5b74d [RPC INTERFACE EVENT]              b25a52bf-e5dd-4f4a-aea6-8ca7272a0e86
1  EFS         START SERVICE CUSTOM                  Microsoft-Windows-EFSTriggerProvider                                    <NULL>
1  hidserv     START SERVICE DEVICE INTERFACE ARRIVAL 4d1e55b2-f16f-11cf-88cb-001111000030 [INTERFACE CLASS GUID]            HID_DEVICE_UP:000C_U:0001
1  gpsvc       START SERVICE NETWORK EVENT           bc90d167-9470-4139-a9ba-be0bbbf5b74d [RPC INTERFACE EVENT]              2EB08E3E-639F-4fba-97B1-14F878961076
1  wudfsvc     START SERVICE CUSTOM                  Microsoft-Windows-DriverFrameworks-UserMode                             4b86f95029922e4ab61551ab3ab10030
2  W32Time     STOP SERVICE  DOMAIN JOINED STATUS     ddaf516e-58c2-4866-9574-c3b615d42ea1 [NOT DOMAIN JOINED]                <NULL>
2  PolicyAgent START SERVICE FIREWALL PORT EVENT     b7569e07-8421-4ee0-ad10-86915afdad09 [PORT OPEN]                        RPC;TCP;%windir%\system32\svchost.exe;policyagent;
2  W32Time     START SERVICE DOMAIN JOINED STATUS    1ce20aba-9851-4421-9430-1ddeb766e809 [DOMAIN JOINED]                    <NULL>
2  lmhosts     STOP SERVICE  IP ADDRESS AVAILABILITY cc4ba62a-162e-4648-847a-b6bdf993e335 [NO IP ADDRESS AVAILABLE]          <NULL>
2  lmhosts     START SERVICE IP ADDRESS AVAILABILITY 4f27f2de-14e2-430b-a549-7cd48cbc8245 [FIRST IP ADDRESS AVAILABLE]       <NULL>
2  IKEEXT      START SERVICE FIREWALL PORT EVENT     b7569e07-8421-4ee0-ad10-86915afdad09 [PORT OPEN]                        500;UDP;%windir%\system32\svchost.exe;IKEEXT;
2  Dnscache    START SERVICE FIREWALL PORT EVENT     b7569e07-8421-4ee0-ad10-86915afdad09 [PORT OPEN]                        5355;UDP;
2  AppIDSvc    START SERVICE CUSTOM                  Microsoft-Windows-AppIDServiceTrigger                                   <NULL>
2  AeLookupSvc START SERVICE CUSTOM                  Microsoft-Windows-ApplicationExperience-LookupServiceTrigger            <NULL>

Statistics:
-----------
Elements processed: 24
Elements output:    15
Execution time:     0.01 seconds
```

Click for larger image

I have Service Trigger data from a few thousand machines, but I'm not at liberty to share it here, trust me when I say finding outliers is easier with a larger data set, but keep in mind, just because something is an outlier doesn't mean it's bad and the inverse is also true, just because something is common, it's not necessarily good.

There is one more ASEP that I know of that Autoruns won't catch, but that Kansa collects, but I'll save that for another post.
