Kansa

Overview

Kansa is a modular, PowerShell based, incident response framework. Kansa utilizes PowerShell Remoting to execute extensible data collection modules across many hosts. Kansa comes preloaded with a variety of modules to help incident responders collect critical forensic data from hosts at scale. The framework also comes with PowerShell scripts to help the responder analyze the data and perform frequency analysis.

“malware artifacts tend to be relatively unique across a file system or an enterprise. By focusing on those outliers, you can often quickly identify a malicious DLL or registry key” (Tilbury, 2010).

Requirements

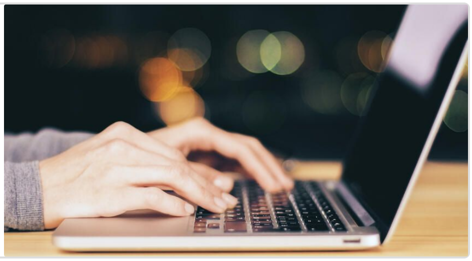
In a domain environment, I have tested using Kansa under a user who has Local Admin privileges set in GPO. We do not need a Domain Admin to run Kansa against a Domain Controller.

PowerShell Remoting should also be enabled throughout the Domain. See the link below for instructions on how to enable PowerShell Remoting via GP.

How to enable PowerShell Remoting via Group Policy

Enabling PowerShell Remoting using Group Policy provides command-level access to all clients, allowing administrators to fully manage devices as if they were sitting at the console locally. Must-

https://www.techrepublic.com/article/how-to-enable-powershell-remoting-via-group-policy/



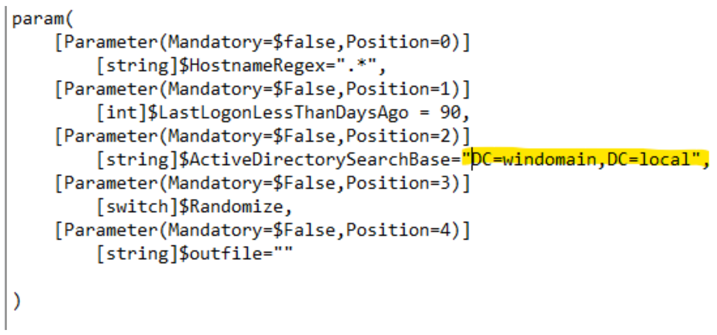
Finally, it is required that Windows Remote Management (WinRM) is enabled. This should be enabled by default unless disabled by the organization.

Installation

Download Kansa - https://github.com/davehull/Kansa

Once installed, run the Prepare-KansaServer.ps1 PowerShell script in the base directory. This script checks to see if Windows Remote Management (WinRM) is running (script will start WinRM if the service is available and not running), checks to see if WinRM listener is enabled, checks to see if the WinRM plugin is available, as well as sets resources constrains.

After the Prepare-KansaServer script is ran, we can run Get-Targets.ps1 > hosts.txt to populate a list of hosts to run Kansa against (note: before running Get-Targets.ps1, specify the domain in the $ActiveDirectorySearchBase variable as shown below (e.g. is for domain name = windomain.local))



```
param(
    [Parameter(Mandatory=$false,Position=0)]
        [string]$HostnameRegex=".*",
    [Parameter(Mandatory=$False,Position=1)]
        [int]$LastLogonLessThanDaysAgo = 90,
    [Parameter(Mandatory=$False,Position=2)]
        [string]$ActiveDirectorySearchBase="DC=windomain,DC=local",
    [Parameter(Mandatory=$False,Position=3)]
        [switch]$Randomize,
    [Parameter(Mandatory=$False,Position=4)]
        [string]$outfile=""

)
```

This script will query active directory for a list of all available hosts to run Kansa against. Alternatively, you can get a list of hosts from the customer and add the FQDN to the hosts.txt file manually.

Important: If you don’t use a host file, Kansa will query the Domain Controller for all available hosts and servers and run triage collection scripts against every device returned.

Usage

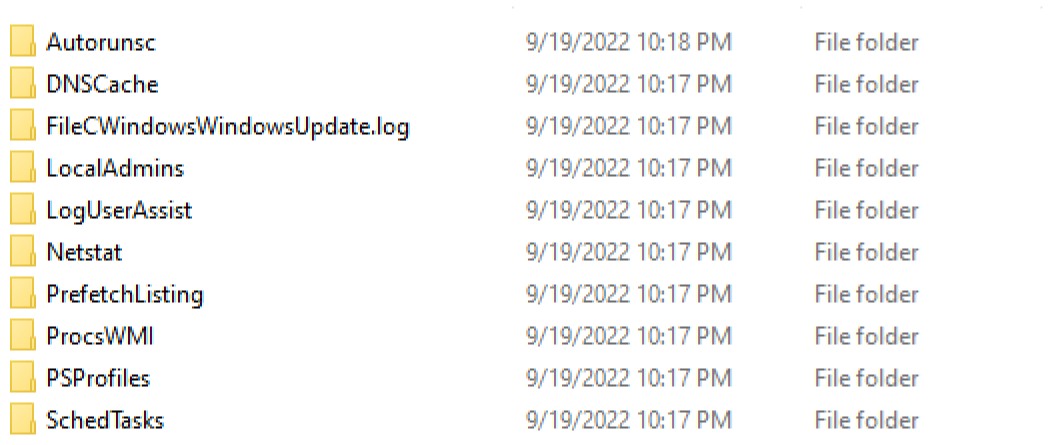
Once installed, use notepad to configure what modules you want to run by editing the Modules.conf page in .\Modules. The modules that Kansa runs will be run the order that they appears in the Modules.conf page. By default, the modules are ran in accordance to the Order of Volatility. Modules that require a third party binary are ran last, as they require the binary to be pushed to the device being investigated and can take longer.

Some modules, such as Get-Autorunsc.ps1 (Kansa\Modules\ASEP) require 3rd party binaries to be pushed to the investigated devices. Place binaries to be pushed in the Kansa\Modules\Bin folder. Then use the -Pushbin flag when running Kansa.ps1 to push the binaries, and -Removebin flag to remove the binary. The -Pushbin flag will push the binaries to the investigated device's C:\Windows folder. Alternatively, consider not adding the -Removebin flag if you think you'll need to run the binary again in a future investigation.

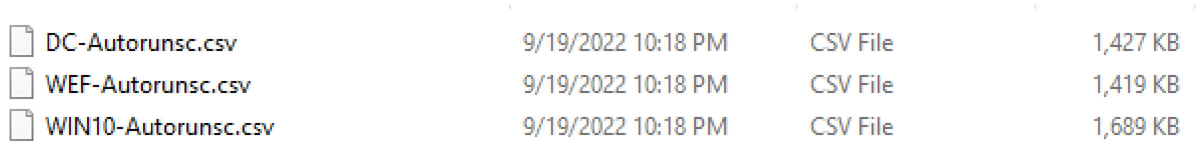
Example Usage:

```
.\kansa.ps1 -TargetList hosts.txt -Pushbin -Verbose
```

A master output folder is generated in the Kansa base directory for each job, with a subfolder for each module run, and a csv for each host containing the collection data.



Autorunsc	9/19/2022 10:18 PM	File folder
DNSCache	9/19/2022 10:17 PM	File folder
FileCWindowsWindowsUpdate.log	9/19/2022 10:17 PM	File folder
LocalAdmins	9/19/2022 10:17 PM	File folder
LogUserAssist	9/19/2022 10:17 PM	File folder
Netstat	9/19/2022 10:17 PM	File folder
PrefetchListing	9/19/2022 10:17 PM	File folder
ProcsWMI	9/19/2022 10:17 PM	File folder
PSProfiles	9/19/2022 10:17 PM	File folder
SchedTasks	9/19/2022 10:17 PM	File folder

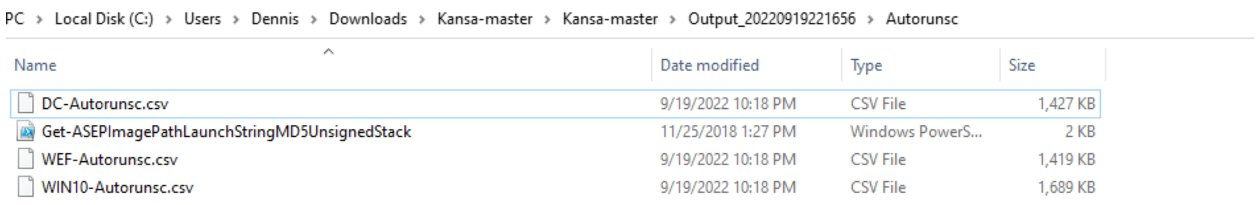


DC-Autorunsc.csv	9/19/2022 10:18 PM	CSV File	1,427 KB
WEF-Autorunsc.csv	9/19/2022 10:18 PM	CSV File	1,419 KB
WIN10-Autorunsc.csv	9/19/2022 10:18 PM	CSV File	1,689 KB

Analysis

Once you have your output, you might want to preform frequency analysis against all of the output pulled from all of the devices. As noted in the beginning of the notion article, malware artifacts should be rare and unique, causing them to stick out. Frequency analysis helps us parse through the forensics artifacts captured in our Kansa output to find outlying indicators. The scripts that we will use for frequency analysis are found in the \Kansa\Analysis folder, and merroir those founds in the Modules folder.

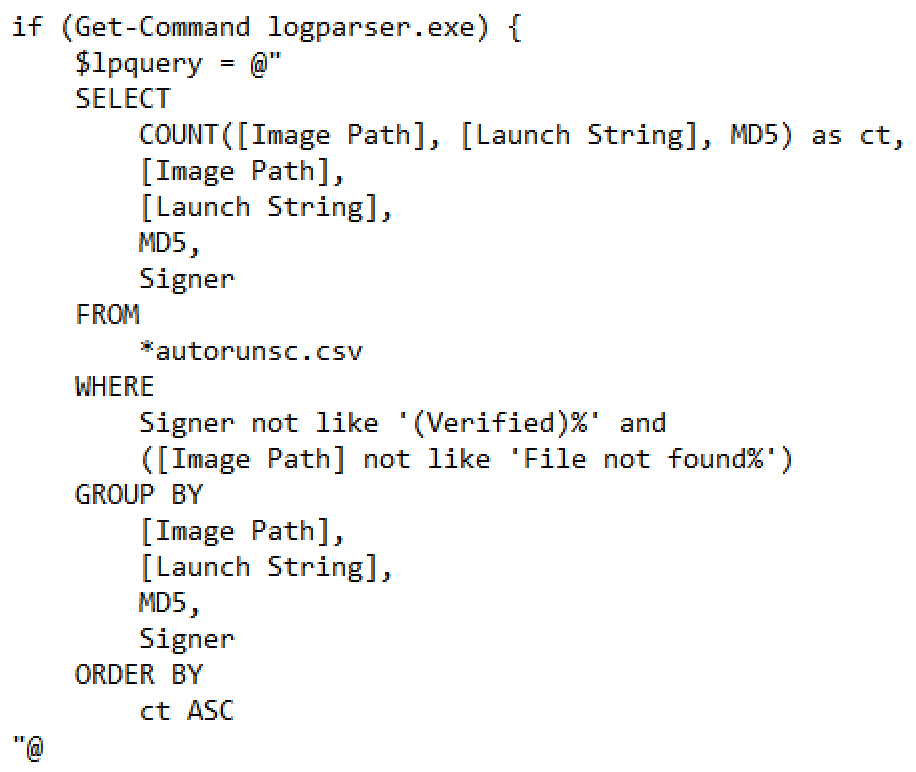
Some of the analysis scripts require Microsoft's LogParser to be added to the path. Once you have LogParser added to the system path, place one of the Analysis scripts in its corresponding outputs directory, and run the Analysis Script



PC › Local Disk (C:) › Users › Dennis › Downloads › Kansa-master › Kansa-master › Output_20220919221656 › Autorunsc

Name	Date modified	Type	Size
DC-Autorunsc.csv	9/19/2022 10:18 PM	CSV File	1,427 KB
Get-ASEPImagePathLaunchStringMD5UnsignedStack	11/25/2018 1:27 PM	Windows PowerS...	2 KB
WEF-Autorunsc.csv	9/19/2022 10:18 PM	CSV File	1,419 KB
WIN10-Autorunsc.csv	9/19/2022 10:18 PM	CSV File	1,689 KB

In this example we will observe Get-ASEPImagePathLaunchStringMD5UnsignedStack.ps1

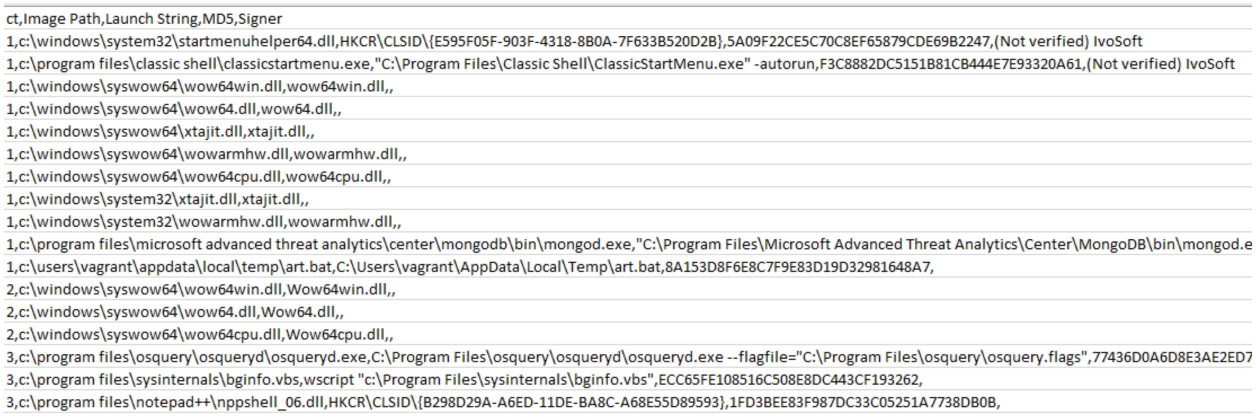


```
if (Get-Command logparser.exe) {
    $lpquery = @"
    SELECT
        COUNT([Image Path], [Launch String], MD5) as ct,
        [Image Path],
        [Launch String],
        MD5,
        Signer
    FROM
        *autorunsc.csv
    WHERE
        Signer not like '(Verified)%' and
        ([Image Path] not like 'File not found%')
    GROUP BY
        [Image Path],
        [Launch String],
        MD5,
        Signer
    ORDER BY
        ct ASC
"@
```

By reviewing the code, we can see that the script is using a combination of LogParser/SQL to count the Image Path, Launch String, and MD5, where the signer is not verified and the image path is not “file not found”, from the any file in the Autorunsc output directory ending in autorunsc.csv.

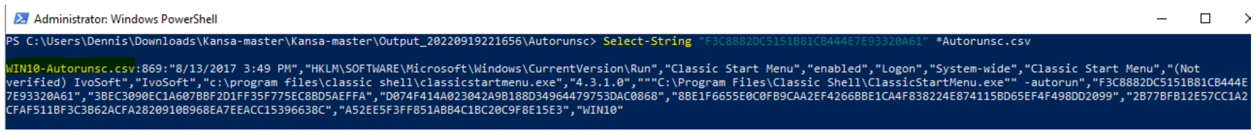
If we run .\Get-ASEPImagePathLaunchStringMD5UnsignedStack.ps1 > Autoruns.csv we can format the output of this script into a csv to observe in Excel/Timeline Explorer.

We can now get an idea of unique occurrences of auto-stat extensibility points.



```
ct,Image Path,Launch String,MD5,Signer
1,c:\windows\system32\startmenuhelper64.dll,HKCR\CLSID\{E595F05F-903F-4318-8B0A-7F633B520D2B},5A09F22CE5C70C8EF65879CDE69B2247,(Not verified) IvoSoft
1,c:\program files\classic shell\classicstartmenu.exe,"C:\Program Files\Classic Shell\ClassicStartMenu.exe" -autorun,F3C8882DC5151B81CB444E7E93320A61,(Not verified) IvoSoft
1,c:\windows\syswow64\wow64win.dll,wow64win.dll,,
1,c:\windows\syswow64\wow64.dll,wow64.dll,,
1,c:\windows\syswow64\xtajit.dll,xtajit.dll,,
1,c:\windows\syswow64\wowarmhw.dll,wowarmhw.dll,,
1,c:\windows\syswow64\wow64cpu.dll,wow64cpu.dll,,
1,c:\windows\system32\xtajit.dll,xtajit.dll,,
1,c:\windows\system32\wowarmhw.dll,wowarmhw.dll,,
1,c:\program files\microsoft advanced threat analytics\center\mongodb\bin\mongod.exe,"C:\Program Files\Microsoft Advanced Threat Analytics\Center\MongoDB\bin\mongod.e
1,c:\users\vagrant\appdata\local\temp\art.bat,C:\Users\vagrant\AppData\Local\Temp\art.bat,8A153D8F6E8C7F9E83D19D32981648A7,
2,c:\windows\syswow64\wow64win.dll,Wow64win.dll,,
2,c:\windows\syswow64\wow64.dll,Wow64.dll,,
2,c:\windows\syswow64\wow64cpu.dll,Wow64cpu.dll,,
3,c:\program files\osquery\osqueryd\osqueryd.exe,C:\Program Files\osquery\osqueryd\osqueryd.exe --flagfile="C:\Program Files\osquery\osquery.flags",77436D0A6D8E3AE2ED7
3,c:\program files\sysinternals\bginfo.vbs,wscript "c:\Program Files\sysinternals\bginfo.vbs",ECC65FE108516C508E8DC443CF193262,
3,c:\program files\notepad++\nppshell_06.dll,HKCR\CLSID\{B298D29A-A6ED-11DE-BA8C-A68E55D89593},1FD3BEE83F987DC33C05251A7738DB0B,
```

We can further pinpoint which device the unique occurrences was generated from by using PowerShell's select-string in the output/ASEP directory (the directory we ran the analysis script from):



```
PS C:\Users\Dennis\Downloads\Kansa-master\Kansa-master\Output_20220919221656\Autorunsc> Select-String "F3C8882DC5151B81CB444E7E93320A61" *Autorunsc.csv

WIN10-Autorunsc.csv:869:"8/13/2017 3:49 PM","HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run","Classic Start Menu","enabled","Logon","System-wide","Classic Start Menu","(Not
verified) IvoSoft","IvoSoft","c:\program files\classic shell\classicstartmenu.exe","4.3.1.0","""C:\Program Files\Classic Shell\ClassicStartMenu.exe"" -autorun","F3C8882DC5151B81CB444E
7E93320A61","3BEC3090EC1A607BBF2D1FF35F775EC8BD5AEFFA","D074F414A023042A9B188D34964479753DAC0868","8BE1F6655E0C0FB9CAA2EF4266BBE1CA4F838224E874115BD65EF4F498DD2099","2B77BFB12E57CC1A2
CFAF511BF3C3B62ACFA2820910B968EA7EEACC15396638C","A52EE5F3FF851ABB4C1BC20C9F8E15E3","WIN10"
```

Limitations

Dave Hull points out that some critics argue that Kansa and more generally, Windows application programming interfaces, could be "subverted" by malware . The results returned by tools that leverage Windows APIs may be falsified. Dave makes no arguments. One method of subverting this limitation is using the *Get-RekalPslist.ps1* module. This module requires the push of Rekall (http://www.rekall-forensic.com/about.html) binary to the system (17mb zipped/35mb unzipped). Once pushed, the script loads winpmem.sys driver, which gives access to Ring0 and then calls the PSlist module to acquire information about processes in memory, including recently exited or unlinked processes.

It is important to note that running Kansa against a system will leave behind artifacts. It's important to know what artifacts are left behind (registry keys, prefetch, ect…) in order to properly attribute artifacts generated by Kansa vs an adversary.