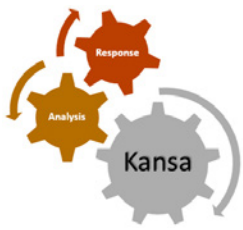
Kansa vs. Cleaver – PowerShell IR Tactics



By Russ McRee – ISSA Senior Member, Puget Sound (Seattle) Chapter

Prerequisites

Windows operating system with Windows Management Framework (includes PowerShell) 4.0. WMF 2.0 and 3.0 work, but 4.0 is recommended.

First of all, Happy New Year! I'm looking forward to a great 2015 and have really appreciated your readership and support during the 2014 schedule.

I am both proud and humbled to announce that this is the *ISSA Journal's* 100th *toolsmith*, and 100 consecutive columns at that. It's really hard to think back to October 2006 and imagine what *toolsmith* would become; it's helped shape my career, my personal philosophy, and I believe it has contributed to the improvement of information security practices for numerous individuals and organizations. Nothing makes me happier than hearing from readers with success stories and wins using the numerous and invaluable tools we've discussed on these pages.

To that end, I'm pleased to cover Kansa[1] for this 100th *toolsmith*. In his own words, Dave Hull's Kansa is a modular framework written in PowerShell for doing incident response (IR) data collection, analysis, and remediation that takes advantage of Windows Remote Management in order to scale up to thousands of systems. It evolved from a few PowerShell scripts that Dave had written for IR work. Through trial and error and some advice from Lee Holmes (@Lee_Holmes), Dave was able to convert those old scripts into a tool that relies on network logons using PowerShell's default non-delegated Kerberos authentication. This ensures that the incident responder's credentials aren't as exposed to harvesting by common adversary tooling such as Mimikatz. This assumes one-hop scenarios (work station to server). In two-hop scenarios, technically not supported by Kansa, (workstation to server to server) the risk remains, as dictated by the necessity for CredSSP, where PowerShell performs a "network clear-text logon." Understand the risks and pitfalls[2] and stick to one-hop scenarios easily supported by Kansa with its Target parameter, where you can run against numerous systems from a single list.

Per Dave, "Kansa is frequently used across enterprises to investigate thousands of systems relatively quickly. It can do lots of great things on its own, but also has the ability to push third-party binaries to remote hosts so more sophisticated tooling can be brought to bear—consider the Get-RekallPslist.ps1 module that installs the WinPMem kernel mode driver on remote hosts and uses it to collect the list of running processes from a live system similar to the way Volatility does against a collected memory image, completely bypassing the Windows API. Because it's written in PowerShell, extending existing modules or creating new ones is relatively easy and Kansa can even be used to clean up systems for tactical containment and remediation."

Dave's written his own excellent Kansa overview and usage guide[3] for PowerShell Magazine, which is a true "must read" for you—do nothing else before doing so. We'll take a more tactical, scenario-based approach to our use of Kansa here so as to provide a different perspective. To that end, Stuart McClure's Cylance group recently published their Operation Cleaver (#OPCLEAVER) report,[4] also a must-read, an in-depth study of an Iranian hacker collective's tools, tactics, and procedures (TTPs). A number of indicators of compromise (IOCs) are included in the report, amongst which are included MD5 hashes for specific malware types including TinyZBot and Lagulon. As described in the report, there are numerous references to "cleaver" inside the namespaces of the collective's custom code, including TinyZBot, thus the name Operation Cleaver. For the best reference material specific to TinyZBot in the report, see the "Persistence" section starting on page 47. We'll focus on using Kansa to find one of the Lagulon keylogging samples.

Tuning Kansa

Kansa is really meant for use across many systems; this one-target scenario is a bit trite on my part. You'll need to imagine the horsepower you see working for this one compromised host across hundreds or thousands of similarly buggered hosts.

On your target servers, script execution needs to be enabled for Kansa use. If you're not familiar with Powershell, that's as easy as:

1. Right-click the Windows PowerShell shortcut.
2. Select *Run* as administrator.
3. Choose *Yes* when the UAC window appears.

1 https://github.com/davehull/Kansa.

2 http://www.powershellmagazine.com/2014/03/06/accidental-sabotage-beware-of-credssp/.

3 http://www.powershellmagazine.com/2014/07/18/kansa-a-powershell-based-incident-response-framework/.

4 http://www.cylance.com/assets/Cleaver/Cylance_Operation_Cleaver_Report.pdf.

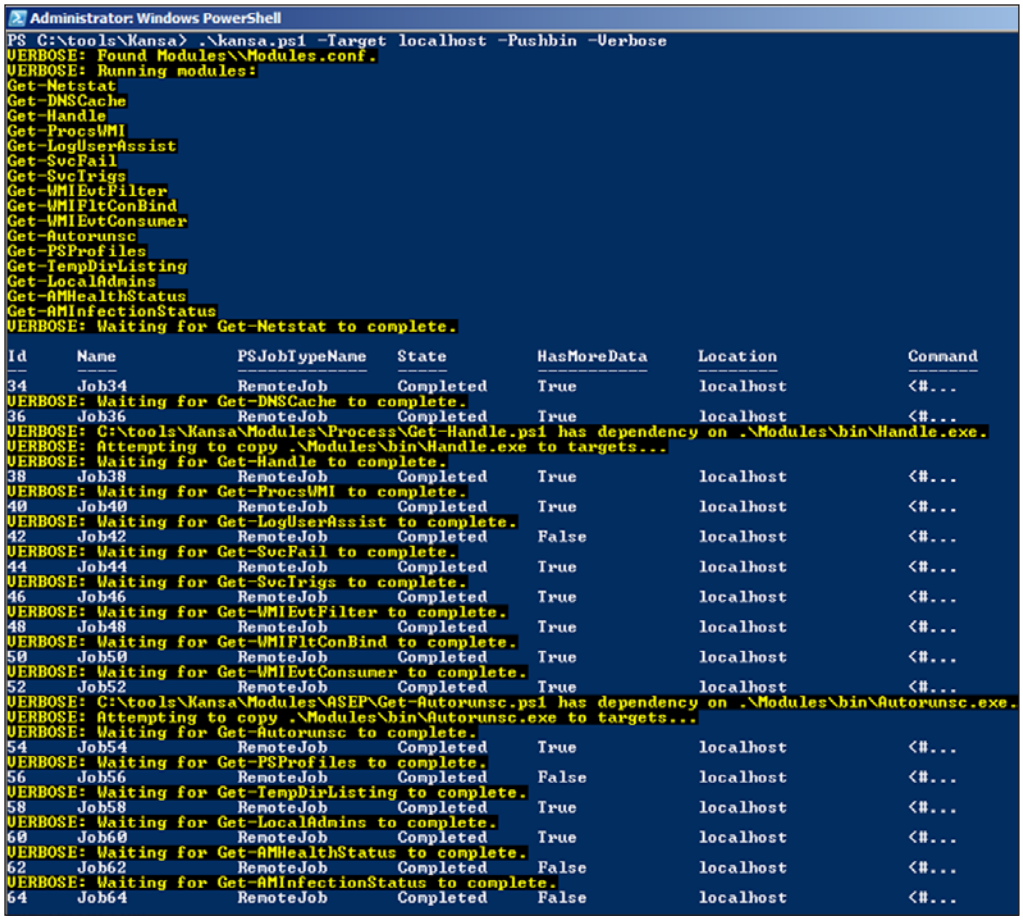


Figure 1 – Kansa test run

4. Use the Set-ExecutionPolicy cmdlet; enter and execute *Set-ExecutionPolicy unrestricted*.
5. You'll also want to run `ls -r *.ps1 | Unblock-File` to unblock the scripts.

You may also need to execute *winrm quickconfig* and/or *Register-PSSessionConfiguration -Name Microsoft.PowerShell* to overcome remote connection errors you may see written to the Kansa output directory if it can't connect to the target host. You may also need to work around Windows firewall issues for PowerShell remoting if any of your network connection profiles are set to Public. You can run *Get-NetConnectionProfile* on Windows 8 or Server 2012 and later to see how your connections are configured, then run the likes of *Set-NetConnectionProfile -InterfaceIndex 25 -NetworkCategory Private* to reset the culprit to Private. You'll modify *–InterfaceIndex* to the correct number matching your Public connection(s).

You should have read Dave's PowerShell Magazine article already, but I will remind you that any binaries you may wish to utilize on target hosts, such as autoruns.exe or handle.exe from the Sysinternals collection, should be stored in the *.\Modules\bin* directory. Beware possible EULA issues with the Sysinternals tools. Even though the Kansa Get-Handle script passes *handle.exe /accepteula –a, handle.exe* hung on me in my test scenario.

A successful pre-infection Kansa run on my intended victim system, as spawned by *.\kansa.ps1 -Target localhost -Pushbin –Verbose*, is seen in figure 1.

Time	EntryLocation	Entry	Enabled	Category	Description	Publi
12/19/2014 20:38	HKLM\System\CurrentControlSet\Control\Session Manager\BootExecute			Boot Execut	System-wide	
12/19/2014 20:38	HKLM\System\CurrentControlSet\Control			Boot Execut	System-wide	
12/19/2014 17:44	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options			Hijacks	System-wide	
12/19/2014 17:44	HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Image File Execution Options			Hijacks	System-wide	
12/19/2014 20:42	HKLM\System\CurrentControlSet\Services			Services	System-wide	
12/19/2014 20:42	HKLM\System\CurrentControlSet\Services			Drivers	System-wide	
12/19/2014 20:38	HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Authentication Packages			LSA Provide	System-wide	
12/19/2014 20:38	HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Notification Packages			LSA Provide	System-wide	
12/19/2014 20:38	HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages			LSA Provide	System-wide	
12/19/2014 20:38	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon			Logon	System-wide	
12/19/2014 20:38	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon			Logon	System-wide	
12/19/2014 20:38	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon			Logon	System-wide	
12/19/2014 17:45	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run			Logon	System-wide	
12/19/2014 17:45	HKLM\Software\Classes*\ShellEx\ContextMenuHandlers			Explorer	System-wide	
12/19/2014 17:45	HKLM\Software\Wow6432Node\Classes*\ShellEx\ContextMenuHandlers			Explorer	System-wide	
12/19/2014 17:45	HKLM\Software\Classes\Drive\ShellEx\ContextMenuHandlers			Explorer	System-wide	
12/19/2014 17:45	HKLM\Software\Wow6432Node\Classes\Drive\ShellEx\ContextMenuHandlers			Explorer	System-wide	
12/19/2014 17:45	HKLM\Software\Classes\Directory\ShellEx\ContextMenuHandlers			Explorer	System-wide	
12/19/2014 17:45	HKLM\Software\Wow6432Node\Classes\Directory\ShellEx\ContextMenuHandlers			Explorer	System-wide	
12/19/2014 20:36	C:\Users\malman\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup	adbReport.exe	enabled	Logon	ecpc\malman	Adob

Figure 2 – Kansa uncovers an autorun entry

ProcessName	ProcId	HandleId	Owner	Type	Perms	Name
adbReport.exe	3396	0x4	ecpc\malman	Key		HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options
adbReport.exe	3396	0x8	ecpc\malman	Directory		\KnownDlls
adbReport.exe	3396	0xC	ecpc\malman	Directory		\KnownDlls32
adbReport.exe	3396	0x10	ecpc\malman	File	(RW-)	C:\Windows
adbReport.exe	3396	0x14	ecpc\malman	Key		HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options
adbReport.exe	3396	0x18	ecpc\malman	Directory		\KnownDlls32
adbReport.exe	3396	0x1C	ecpc\malman	File	(RW-)	C:\Windows\SysWOW64
adbReport.exe	3396	0x20	ecpc\malman	Key		HKLM\SYSTEM\ControlSet001\Control\Nls\Sorting\Versions
adbReport.exe	3396	0x24	ecpc\malman	Key		HKLM\SYSTEM\ControlSet001\Control\SESSION MANAGER
adbReport.exe	3396	0x28	ecpc\malman	ALPC		Port
adbReport.exe	3396	0x38	ecpc\malman	Key		HKLM
adbReport.exe	3396	0x48	ecpc\malman	WindowStation		\Sessions\1\Windows\WindowStations\WinSta0
adbReport.exe	3396	0x4C	ecpc\malman	Desktop		\Default
adbReport.exe	3396	0x50	ecpc\malman	WindowStation		\Sessions\1\Windows\WindowStations\WinSta0
adbReport.exe	3396	0x54	ecpc\malman	Key		HKLM\SYSTEM\ControlSet001\Control\Nls\CustomLocale
adbReport.exe	3396	0x8C	ecpc\malman	Mutant		\Sessions\1\BaseNamedObjects\Adobe Report Service
adbReport.exe	3396	0x90	ecpc\malman	Directory		\Sessions\1\BaseNamedObjects
adbReport.exe	3396	0x94	ecpc\malman	Section		\Sessions\1\BaseNamedObjects\windows_shell_global_counters
adbReport.exe	3396	0x98	ecpc\malman	Key		HKLM\SYSTEM\ControlSet001\Control\Nls\Locale\Alternate Sorts
adbReport.exe	3396	0xB8	ecpc\malman	Key		HKLM\SYSTEM\ControlSet001\Control\Nls\Locale
adbReport.exe	3396	0xBC	ecpc\malman	Key		HKLM\SYSTEM\ControlSet001\Control\Nls\Language Groups
adbReport.exe	3396	0xC8	ecpc\malman	ALPC		Port
adbReport.exe	3396	0xE0	ecpc\malman	File	(---)	\Device\KsecDD
adbReport.exe	3396	0xE4	ecpc\malman	Mutant		\Sessions\1\BaseNamedObjects\MSCTF.Asm.MutexDefault1
adbReport.exe	3396	0xE8	ecpc\malman	ALPC		Port
adbReport.exe	3396	0xF0	ecpc\malman	File	(R-D)	C:\Windows\SysWOW64\en-US\msctf.dll.mui
adbReport.exe	3396	0x10C	ecpc\malman	ALPC		Port
adbReport.exe	3396	0x124	ecpc\malman	Key		HKCU
adbReport.exe	3396	0x128	ecpc\malman	Key		HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings
adbReport.exe	3396	0x12C	ecpc\malman	Key		HKU
adbReport.exe	3396	0x130	ecpc\malman	Key		HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer
adbReport.exe	3396	0x178	ecpc\malman	File	(RW-)	C:\Users\malman\AppData\Local\Microsoft\Windows\Temporary Internet Files\counters.dat
adbReport.exe	3396	0x188	ecpc\malman	Key		HKLM\SOFTWARE\Wow6432Node\Microsoft\Internet Explorer\MAIN\FeatureControl
adbReport.exe	3396	0x18C	ecpc\malman	Key		HKLM\SOFTWARE\Policies\Microsoft\Windows\CurrentVersion\Internet Settings
adbReport.exe	3396	0x190	ecpc\malman	Key		HKCU\Software\Policies\Microsoft\Windows\CurrentVersion\Internet Settings
adbReport.exe	3396	0x194	ecpc\malman	Key		HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings

Figure 3 – Kansa pivots on adbreport.exe handles

Kansa vs. Cleaver

I acquired a few samples, as described in Cylance's Operaration Cleaver report, from VirusShare.com, as always my favorite repository. After investigating the run-time behaviors of a few of them, I found a specific sample, a Lagulon variant, that was a good representative for Kansa use as it hit marks for a few of the IOCs mentioned in the report. The sample, MD5: 53230e7d5739091a6eb51298a50eb616,[5] is discussed generally on page 58 in the report as part of the wndTest analysis, observed attempting to impersonate Adobe Report Service.

After executing the sample on my victim system (I had to pwn a real Windows image as the malware didn't run in a VM environment), I re-ran Kansa and collected results from the Output directory. I then went on a subsequent search for all things Adobe related and uncovered more than a few IOCs. As all output files are TSV by default, they play nicely in Excel. The first hit came from the Get-Autorunsc module, the results from which I filtered by infection date to reduce the noise. The output as seen in figure 2 shows that adbreport.exe has been configured to run at logon via C:\Users\malman\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup.

The next hit came from the Get-Handle module, the results from which I sorted alphabetically by Process Name to key on adbReport.exe. As you can see in figure 3, adbReport.exe, as identified by Process ID 3396, has open handles on a number of files and registry entries.

Note the \Device\KsecDD entry in figure 3 as well, indicating direct device communication via the KsecDD driver.

More related IOCs were noted via the Get-ProcsWMI module which, per its own synopsis, "acquires various details about running processes, including command-line arguments, process start time, parent process ID, and MD5 hash of the process image as found on disk," the results of which you can see in figure 4.

You've likely come to the conclusion that the hash referenced in figure 4 is in fact that associated with the malicious binary acquired from VirusShare, thus closing the circle of discovery and incident enumeration with Kansa.

That said, again, single host scenarios with Kansa are cute, but it's made to shine at scale. Viewing individual TSV files in Excel is definitely not a scale solution. How about viewing and parsing results via Log Parser? Better, right? Integrate with SQL-based storage and you're really off to the races. The Kansa Analysis folder includes a number of analysis-centric scripts to work with the results. They're most often frequency-analysis oriented to work across a body of results from a

5 https://www.virustotal.com/en/file/c11a244cba9da30173ff1dcb755a377c3b2b1f99cd15a887041937b086113ebd/analysis/.

Hash	ProcessName	Handles	VM	WS	Path
53230E7D5739091A6EB51298A50EB616	adbReport.exe	141	82104320	7491584	C:\Users\malman\AppData\Roaming\Mi
47EA5F76FAB723C61AB4A0D79BAD512C	AdobeARM.exe	289	107876352	13185024	C:\Program Files (x86)\Common Files\Ac
B362181ED3771DC03B4141927C80F801	armsvc.exe	75	44851200	4030464	C:\Program Files (x86)\Common Files\Ac
D5CCA1453B98A5801E6D5FF0FF89DC6C	audiodg.exe	157	69210112	17870848	C:\Windows\system32\AUDIODG.EXE
5746BD7E255DD6A8AFA06F7C42C1BA41	cmd.exe	20	39182336	2760704	C:\Windows\system32\cmd.exe
BF95EA5809E3BBF55370F7CB309FEBD0	conhost.exe	60	61386752	6463488	C:\Windows\system32\conhost.exe
BF95EA5809E3BBF55370F7CB309FEBD0	conhost.exe	58	60076032	5521408	C:\Windows\system32\conhost.exe
60C2862B4BF0FD9F582EF344C2B1EC72	csrss.exe	637	47300608	4517888	C:\Windows\system32\csrss.exe
60C2862B4BF0FD9F582EF344C2B1EC72	csrss.exe	322	58998784	10334208	C:\Windows\system32\csrss.exe

Figure 4 – Kansa closes the circle with the adbreport.exe MD5 hash

```
40  wininit.exe              NT AUTHORITY\SYSTEM
41  igfxpers.exe             ecpc\malman
41  lsm.exe                  NT AUTHORITY\SYSTEM
48  UNS.exe                  NT AUTHORITY\SYSTEM
48  adbReport.exe            ecpc\malman
50  HPTaskBar2.exe           ecpc\malman
50  HPTaskBar1.exe           ecpc\malman
52  taskeng.exe              ecpc\malman
53  SearchProtocolHost.exe   ecpc\malman
54  conhost.exe              ecpc\malman
55  IAStorDataMgrSvc.exe     NT AUTHORITY\SYSTEM
59  spoolsv.exe              NT AUTHORITY\SYSTEM
59  winlogon.exe             NT AUTHORITY\SYSTEM
77  HPSA_Service.exe         NT AUTHORITY\SYSTEM
80  SynTPEnh.exe             ecpc\malman
87  taskhost.exe             ecpc\malman
94  wsmprovhost.exe          ecpc\malman
98  MSOIDSVC.EXE             NT AUTHORITY\SYSTEM
109 AdobeARM.exe             ecpc\malman
119 services.exe             NT AUTHORITY\SYSTEM
120 powershell.exe           ecpc\malman
126 csrss.exe                NT AUTHORITY\SYSTEM
139 wmpnetwk.exe             NT AUTHORITY\NETWORK SERVICE
183 SearchIndexer.exe        NT AUTHORITY\SYSTEM
223 lsass.exe                NT AUTHORITY\SYSTEM
266 svchost.exe              NT AUTHORITY\NETWORK SERVICE
297 explorer.exe             ecpc\malman
397 System                   \<unable to open process>
486 svchost.exe              NT AUTHORITY\LOCAL SERVICE
639 svchost.exe              NT AUTHORITY\SYSTEM
PS C:\tools\Kansa\Analysis\process>
PS C:\tools\Kansa\Analysis\process>
PS C:\tools\Kansa\Analysis\process> .\Get-HandleProcessOwnerStack.ps1 | findstr "adbReport"
48  adbReport.exe            ecpc\malman
PS C:\tools\Kansa\Analysis\process>
```

Figure 5 – Kansa frequency analysis across results

plethora of hosts. Sadly my examples will only be from my one wee victim system, but you get the point.

As an example in Analysis\Process you can use Get-HandleProcessOwnerStack.ps1 to pull frequency from all collected Get-Handle data based on Process Name and Owner. While I only collected one result set, *.\Get-HandleProcessOwnerStack.ps1 | findstr "adbReport"* tells me that adbReport.exe has 48 handles open as seen in figure 5.

Had there been one or 1000 Get-Handle results in the working directory where Get-HandleProcessOwnerStack.ps1 was run from, it would count and sort ALL handles by process and owner.

If you need a frequency count of all instances of adbReport.exe via an MD5 hash match across all results from all hosts, you need only run *Get-ProcsWMICLIMD5Stack.ps1*. My favorite is sorting the same data set by creation date. This helps while conducting timeline analysis and will tell you when a process was created across all targets, the like of which is seen in figure 6.

Response capabilities, analysis of the results, and even the premise of remediation via Kansa for the adventurous and innovative amongst you, all built into one tidy PowerShell framework. For modern Windows environments, consider Kansa a must have in the IR toolkit.

In conclusion

Come what may, in light of attacks as discussed in the Operation Cleaver report, tool frameworks such as Kansa help you leverage the real Windows workhorse that is PowerShell in order to better respond and analyze. Keep an eye on the Kansa Github site for updates and news and follow @davehull on Twitter. Dave and I also both follow @Lee_Holmes, whom we both consider to be the Don of the PowerShell family.

Remember to vote for your favorite tool of 2014 through January 15, 2015. I'll announce a winner soon thereafter. Please vote and tell your friends and co-workers to do the same.

Ping me via email or Twitter if you have questions (russ at holisticinfosec dot org or @holisticinfosec).

Cheers…until next month.

Acknowledgements

—Dave Hull (@davehull), Kansa developer and project lead

About the Author

Russ McRee manages the Threat Intelligence & Engineering team for Microsoft's Online Services Security & Compliance organization. In addition to toolsmith, he's written for numerous other publications, speaks regularly at events such as DEFCON, Black Hat, and RSA, and is a SANS Internet Storm Center handler. As an advocate for a holistic approach to the practice of information assurance Russ maintains holisticinfosec.org. He serves in the Washington State Guard as the Cybersecurity Advisor to the Washington Military Department. Reach him at russ at holisticinfosec dot org or @holisticinfosec.

```
PS C:\tools\Kansa\Analysis\process> .\Get-ProcsWMISortByCreationDate.ps1 | findstr "adbReport"
20141219203920.218698-480 3396      2640           C:\Users\malman\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\ad
```

Figure 6 – Kansa determines creation dates across results