

# Incident Handling Preparation: Learning Normal with the Kansa PowerShell Incident Response Framework


Jason Simsay

# Incident Handling Preparation

*Learning Normal with the Kansa PowerShell Incident Response Framework*

*GIAC (GCIH) Gold Certification*

Author: Jason Simsay, jason.simsay@gmail.com
Advisor: Mohammed Haron
Accepted: August 11, 2016

Final

Abstract

Preparation is a critical step in establishing an effective incident response program. Information Security professionals that will be called upon to handle an incident must prepare ahead of time. Kansa is a PowerShell Incident Response Framework developed by Dave Hull. The PowerShell Remoting feature is leveraged to establish a highly scalable and extensible system state collection platform. Once data is collected from across the Microsoft environment, an extensive set of frequency analysis scripts may be executed to enable incident handlers to turn unknowns into knowns and to discover anomalies and indicators of compromise.

Learning and deploying the Kansa framework before an incident occurs is invaluable preparatory work. Security analysts can leverage Kansa to baseline systems and establish familiarity with the normal state. Being familiar with normal makes us much more adept at identifying the abnormal, getting a jump on incident identification, and ideally, containing a security incident with minimal damage. This paper will explore leveraging the Kansa framework to facilitate documentation of normal state baselines. We will build upon the frequency analysis capabilities to support profiling homogenous endpoint profiles deployed across the organization.

# 1. Introduction

Your organization is compromised. Go! Is your team caught up in routine event log reviews? Even if you may have narrowly avoided a resume generating event, there remains the lingering data breach issue. External breach notification in the absence of an anomalous event leaves the incident response team and security analysts searching for a needle of compromise in a haystack of event logs and network activity. The 2016 Verizon Data Breach report stated that the detection deficit, the time to compromise versus time to discover, is getting worse (Verizon, 2016). Hackers are getting in faster and organizations are taking longer to discover the attack. If an intrusion is undetected when it first occurs how will we discover it afterward?

Skilled adversaries can blend their activities into what appears to be normal user and administrator activity, making rapid detection even more challenging. Security operations teams must look for known indicators of compromise and also look for the unknown. Both are formidable tasks. Publicly known indicators of compromise are extensive and searching for what you do not know in a modern operating system will most likely be futile. "Live Response Using PowerShell" by Sajeev Nair presents a compiled list of over 30 Windows system artifacts to be searched for potential indicators (Sajeev 2013). Fortunately, on our side is a large dataset comprised of the configuration state of all the hosts on our networks. Extracting the pertinent information across the dataset can provide some semblance of what is normal and what is abnormal. A SANS Digital Forensics and Incident Response (DFIR) blog entry by Chad Tilbury describes Mandiant's Peter Silberman's Least Frequency of Occurrence principal as simple but powerful: "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).

Frequency analysis then is the analysis of the frequency of occurrence of a particular indicator or data point across a dataset. Dave Hull has provided "Kansa: A PowerShell-based incident response framework" to the information security community (Hull, 18 July 2014). Kansa is a framework that leverages the PowerShell Remoting feature to execute extensible data collection modules across many hosts. Dave informs

us that the data must be analyzed and has provided sample scripts which he says "for the most part" perform frequency analysis. Dave described how he was able to leverage Silberman's Least Frequency of Occurrence Principle to reduce the size of a data set in a 2011 SANS DFIR blog post (Hull, 2011). Some describe its usefulness as an awesome method to detect anomalous indicators dropped by polymorphic or metamorphic malware (Hosmer, 2008).

Organizations can leverage those remarkable ideas and the works of these experts to prepare to handle security incidents. SANS Incident Handling Step-by-Step describes preparation as imperative, "to make sure we have the skills and the resources that we need ready to go at a moment's notice" (SANS, p 18). This paper will summarize key incident response strategies, consider the value and risk of an enterprise roll out of PowerShell Remoting capabilities, and guide a secure deployment. With the infrastructure in place, an organization can implement Kansa and leverage the data collection capabilities to become familiar with the normal state of the environment so that the abnormal state can be more readily identified. Kansa's primary use case is during an active incident response engagement. The use case herein is one of preparation, specifically, learning what normal looks like. We will extend the Kansa data collection and analysis functions with mechanisms to group and filter systems and establish baselines for the various system profiles in the environment. Along the way, we will develop familiarity with the tool and learn to customize it. This preparatory process is also a hunt team exercise. There is a chance we might discover the adversary in our midst.

## 2. Incident Handling Preparation

The National Institute of Standards and Technology Special Publication 800-61, Computer Security Incident Handling Guide, states that establishing a strong incident response capability requires substantial planning and resources (NIST, Abstract). There are some significant organizational considerations. We are going to touch on just a few of those most relevant to knowing what is normal and discovering abnormal.

## 2.1. Active and Passive Response

Organizations should determine ahead of time the criteria for determining whether to initiate an active response that will tip off the adversary or to remain passive and merely observe. NIST advises that any plans to observe should include enough containment to sandbox the adversary to avoid potential liabilities (NIST, p. 36). "The goal of live response is to identify incidents as quickly as possible. To do that you want to collect the right information that helps you make the decision" (Nair 2013). If we don't have the necessary information on hand to make a determination of how to handle an active adversary, we want the option and need the capability to find that information quickly.

## 2.2. What is Normal?

Dave Hull states that incident response teams are regularly called to action with very limited knowledge about the incident. Furthermore, the scope of the investigation often expands quickly (Hull, 18 Jul 2014). Incident response teams must strive to familiarize themselves with the normal state of their environment. NIST guidance suggests we study systems and networks to understand normal to more readily identify abnormal (NIST 3.6). Data filtering, or reduction, to ensure time is spent on that which is likely to be most interesting, is also an important concept (NIST, 3.2.4). The Kansa framework and Least Frequency of Occurrence principal are ideally suited to accommodate these preparations.

## 2.3. Know Your Resources

Having the right tools and knowing how to use them is critical to turning the tide on the detection deficit metric defined by Verizon. A study by McAfee found 33% of organizations reported the lack of good tools to baseline normal behavior as a contributor to inadequate security visibility. Thirty-seven percent sought better integration between security intelligence and operational tools (McAfee, 2015). NIST tells us to "Acquire tools and resources that may be of value during incident handling" (NIST 3.6). We will become familiar with PowerShell Remoting and Kansa. Acquiring human resources and establishing relationships with the extended incident response team could be one of the

most important preparations and should not be overlooked. Know the system experts within IT and within the various business units.

### 2.4. Event Detection/Incident Identification

"There is a very pronounced tendency to wait until we are sure something is wrong before we alert" (SANS, p. 51). The paper "Live Response with PowerShell" asks "Do you have the right information available to determine if a security incident has occurred" (Nair, 2013). By collecting the right information before we are in response mode, we can be more certain something is wrong and provide an alert sooner when an incident has occurred. This is assuming that we are fortunate enough to detect the incident before an external party reports it to us. Ultimately, we need to reduce the detection deficit.

## 3. Kansa PowerShell IR Framework

Kansa is available from Dave Hull on GitHub. His description: "It uses PowerShell Remoting to run user contributed, ahem, user contributed modules across hosts in an enterprise to collect data for use during incident response, breach hunts, or for building an environmental baseline" (Hull). He goes into more detail about Kansa, the value of its many modules to incident response, and the analysis scripts in a July 2014 article in PowerShell Magazine. What follows in the next two sections is a brief recap of that article.

### 3.1. Collect Data

Kansa can collect data using PowerShell, standard Windows command line tools, or third party executables. The data collection is defined within a module, and you can run one or many modules against one or tens of thousands of targets. Kansa has been designed to collect the most volatile artifacts first. The order that modules are specified in the modules configuration file is the order in which they will be invoked. An output directory is created for each Kansa run and includes a subdirectory for each module which contains an output file for each target. As of this writing, Kansa includes more than 50 modules to retrieve data from Windows system artifacts known to be altered by the presence of malware or adversaries. PowerShell Remoting provides Kansa with

massively scalable capabilities and is covered below. You can acquire a lot of pertinent data very quickly!

## 3.2. Analyze Data

Frequency analysis is also known as data stacking. The objective is to find an attribute which would be expected to have a common or consistent value across the data set. Group the data on this attribute and determine whether you have 100 or 1 occurrence(s) of that particular value. If you pick something that is unique to every system, you will end up with a grouping that means nothing (Fireeye). The Kansa framework provides us with more than 40 analysis scripts to perform these interesting groupings for us. You may find that you execute a given module and then run multiple analysis scripts against the same data set. Get-LogParserStack.ps1 has been recently added to the GitHub repository and enables interactive flexibility when analyzing the data. The Microsoft LogParser utility is a required prerequisite for running these analysis scripts.

## 3.3. Getting Up and Running

First, ensure you have obtained the appropriate authorization to install new and target organizational assets with new tools. Download the Kansa archive from GitHub, extract it, run the PowerShell Unblock-Files cmdlet, and you are ready to begin collecting data. Start the exercise with an analysis of your local fully qualified domain name or any available Windows 2012 or later servers. The default configuration for more recent server operating systems has Windows Remote Management and the PowerShell Remoting feature enabled by default unless disabled by your organization. Earlier Windows releases will require PowerShell Remoting to be enabled. Verify accessibility of the remote listening service using the Invoke-Command cmdlet. Kansa's modulePath parameter will allow you to specify an individual collection module. As with anything new, it is best to start small and build up as you gain experience. Running Kansa.ps1 with no arguments on a system with Remote Server Administration Tools available will discover all computers in Active Directory and execute many modules against all of them. This is not recommended practice for just getting started. Rather, test each module on a few computers and observe the results.

To be most useful it is necessary to be knowledgeable of the various artifacts the modules will collect. You should be well into Dave's PowerShell Magazine article by now. The article goes well beyond running Kansa and explains the data targeted by the various modules in terms of its value to incident response. Pick and choose those modules which have familiar data and get them working. Decide which third party executables you want to leverage and push them to your targets. Use the analysis parameter to automatically analyze the collected data. You must be sure that your modules and analysis configuration files are aligned. You cannot analyze what you do not collect. If you do not use the analysis parameter, you will have to analyze manually. You can use the provided analysis scripts, the Get-LogParserStack.ps1 scripts, Microsoft LogParser.exe, Excel, or whatever method is comfortable.

Once familiar with the modules, determine what additional information would be useful. Creating module and analysis scripts is a great way to further your familiarity with Kansa. If you put time and effort into developing something the community can benefit from, please contribute.

## 3.4. A GitHub Fork to Get You Up and Running Faster

Determining which modules to use and getting the modules and analysis configuration files in sync takes some effort. Some of the analysis scripts are expecting tab-separated values files whereas the modules return comma-separated values files, necessitating two minor changes to the analysis scripts.

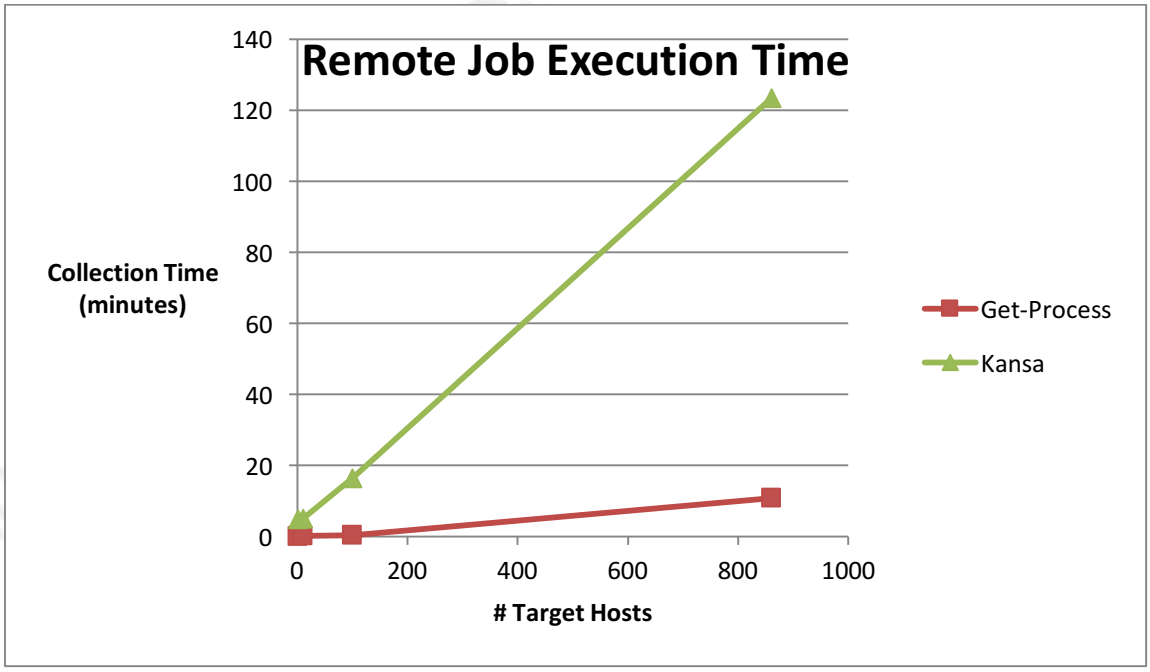
1. The SQL query passed to the Microsoft LogParser tool must select CSV files rather than TSV files
2. The parameters used for the LogParser command must be revised to reflect the CSV input format and drop the fixedsep parameter as it is only valid for the TSV input format

Modified scripts that address these issues are available at https://github.com/dry-fly/Kansa-Profiler. These scripts will get you up and running quicker. The repository also includes additional modules and scripts used for profiling systems as described herein.

# 4. PowerShell Remoting

Chris Hallenbeck's post to the Tanium Blog singles out the "flexibility to run ad-hoc searches across the enterprise for any scope of data – current, at-rest, or historical, and get immediate responses" as a key capability that reduces the time and effort required for a successful response. He elaborates, saying that without rapid responses the team's "ability to innovate" is impeded, thus "forcing reliance on stale, limited sets of data." The result is that investigations take longer to complete and remediate (Hallenbeck, 2016). Kansa provides the framework for this capability and PowerShell Remoting the supporting infrastructure.

What is PowerShell Remoting? It's a feature of PowerShell that is built on the Windows Remote Management (WinRM) framework and enables you to invoke commands that process remotely on one or many hosts. It is highly scalable. The chart below shows the time to run invoke-command to execute get-process and Kansa.ps1 for 1, 10, 100, and 861 remote targets. By default, Kansa throttles to a maximum of 32 simultaneous jobs. The result is a linear relationship. See appendix for additional detail.




## 4.1. Deployment Considerations

PowerShell Remoting creates significant trace activity on the target system and is most definitely an active response (Adams, p 11). Establishing a remote session creates

log entries, runs processes, and modifies user files and registry hives, all of which effectively alter the state of the disk, potentially compromising integrity. This raises concerns with attribution if an organization desires to take legal action against an intruder. Furthermore, it may tip off the intruder. (Adams, 2015)

Robert Adams interviewed Microsoft's Senior Security Analyst Brian Hooper regarding passive versus active response strategies. Hooper states that the decision is made based on the sophistication of your adversary. He suggests that when dealing with a sophisticated adversary, you would want to take the affected system(s) offline (Adams, 2015). When routine PowerShell Remoting processes are established and are running remote commands, the organization is less likely to tip off the adversary. As the incident becomes more understood, additional data collections could certainly help define the exact level of adversarial sophistication. An incident response team must execute actions by the organization's established plans and make risk-based decisions as needed (Adams, 2015).

The use case described herein is that of collecting system state configurations to baseline normal. This is also, at the same time, a hunt team exercise. A few data collection cycles before the adversary appears and the remote PowerShell modifications are now a part of the normal operations on the box. Regardless, ensure that your incident response program supports the use of PowerShell Remoting in both a preparatory and active response capacity.

Dave Hull points out that some critics argue that Kansa and more generally, Windows application programming interfaces, could be “subverted” by malware (Hull, 14 April 2014). The results returned by tools that leverage Windows APIs may be falsified. Dave makes no arguments. It is admittedly something to be aware of but does not detract from the value of using said tools.

## 4.2. Securing WS-Man/Windows Remote Management

Web Services-Management is a standard management protocol and Microsoft’s implementation is Windows Remote Management (WinRM) (Wikipedia). The National Security Agency/Central Security Service Information Assurance Directorate paper, *Spotting the Adversary with Windows Event Log Monitoring,* is an excellent resource for

establishing a Windows Event Log collection infrastructure using the native Windows event collection capabilities (IAD, 2013). Like PowerShell Remoting, Windows Event Logging leverages WinRM services. The paper covers the secure configuration of WinRM in excellent detail with recommendations to harden the service. Read it and use it as you consider deploying WinRM within your environment. A few bullets summarize the considerable value of the document:

- Protect the SVCHOST process with Microsoft's Enhanced Mitigation Experience Toolkit
- Disable all authentication methods except Kerberos, except, leave Negotiate enabled until after you have completed configuration
- Specific requirements are given for environments that have non-domain machines.
- Leverage the Windows Firewall to restrict access to authorized management systems

A few years have elapsed since publication. Robert Adams describes a host of new security features available in recent versions of WinRM and PowerShell (Adams, 2015). These come as organizational red and blue teams and adversarial entities continue to leverage and expand upon the power of these technologies. Monitoring routines for PowerShell transactional logging will identify unauthorized use of the tool.

## 5. Slicing and Dicing the Data collected with Kansa

Frequency analysis of data collected from across an environment is an effective means of discovering unusual artifacts. During incident response preparation, the system baselining use case will require the ability to group systems based on their role in the environment. Servers and desktops and front line and back shop user endpoints have significantly different system configuration states. Varying hardware and operating systems will present different system states. By grouping and filtering based on the variables that define the system profile, and then analyzing the subset, we can get a more concise baseline for systems of that profile.

Is it worth the effort to slice and dice the data? The answer is it depends on what you are looking for. Anomalous artifacts or indicators of compromise might stand out when looking at the entire data set. A rogue end user system with a connection to a backend SQL server, to which a web server farm is connected, might not jump out in an analysis of the entire data set. However, an analysis of only those end user systems would be likely to identify the backend connection to the SQL server.

Do we run multiple Kansa data collections targeting each system profile or do we run a single data collection? A single collection offers the option to analyze the entire data set to identify only far outliers or break it down and analyze each profile to identify more subtle outliers. Kansa, as delivered from Dave Hull's GitHub site, has the analysis of the entire data set covered. To become familiar with the normal state and baseline systems based on profile, we must develop a way to break down the data set.

## 5.1. Breaking it Down: Profile Identification and Isolation

Assets can be grouped on system properties or properties defined by the organization. System properties would include the operating system, architecture, hardware manufacturer, and model. Active Directory (AD) organizational units are likely present in your environment and may serve to group systems of similar functions. The relevant analysis buckets probably already exist in some form within most organizations to support existing reporting processes. Regardless, you will need to determine what properties delineate your target system profiles. The following steps will enable us to identify the normal baseline for each profile.

1. Build additional Kansa modules to collect system properties
2. Enumerate the Kansa target list file and retrieve information from AD and the data collection to create a database of system properties
3. Read the database into a PowerShell object to group and filter and isolate the systems based on profile
4. Establish a profile directory containing links to the Kansa data files

At this point, without duplicating or reacquiring data, we have isolated the system profile of interest from the larger data collection. The isolated profile directory mirrors

the layout of the typical Kansa output directory such that the same analysis processes will apply.

## 5.2. Breaking it Down: Profile Analysis

Kansa has an analysis parameter that will execute the analysis scripts specified in the analysis configuration file. Can we quickly perform this analysis for the isolated profile data directory? The Kansa PowerShell script function Get-Analysis executes when the analysis parameter is specified. We can reuse Get-Analysis and incorporate it into the profiling scripts.

We will accomplish system profiling by running two scripts. First, createProfilingDatabase.ps1 will create the system properties database for our target hosts. Second, createProfileDirectory.ps1 will establish the directory of links to the collected data and enable integration of the Get-Analysis function. Once the systems are profiled, a third script, kansaGet-Analysis.ps1 enables us to reuse the Get-Analysis function and contains the function code from Kansa.ps1 as well as its dependent function Get-Directives. This new PowerShell script does not create any output enabling it to be "dot sourced" in the createProfileDirectory.ps1 script. We must also establish a log file and the parameter variables required by the Get-Analysis function.

## 5.3. Breaking it Down: System Profiling Step by Step

Once you have Kansa working in your environment, you are ready to use the guidance herein to isolate and baseline system profiles of interest. The Get-WMIComputerSystem.ps1 and Get-WMIOperatingSystem.ps1 modules are available on GitHub. Add these to your modules configuration file and conduct another Kansa run. These modules retrieve hardware and operating system properties directly from the target hosts. These properties are required to build the profiling database. This data is collected and stored the same as all other Kansa modules. Optionally, you could just run Kansa with a modules configuration that specifies only these two modules and then copy the output directories into a previous data collection output directory.

### 5.3.1. createProfilingDatabase.ps1

The script references an existing Kansa target list file to build a database of system properties from data collected by Kansa and from Active Directory. We import any Kansa collected system property data into a PowerShell object and also search AD to retrieve attributes from the target computers to include in the database.

Set the kansaPath and kansaHosts variables to reflect your configuration. Run the script run with the Kansa output directory of interest as your current working directory. The script will create a "_Profiles" subdirectory and output is written to the profilingDatabase.CSV file. Pseudo code for the createProfilingDatabase.ps1 script is as follows:

```
Get contents of Kansa targetList file
Populate PowerShell objects with CSV data collected by property modules
For each target from targetList {
        get-adcomputer properties from Active Directory
        populate PSCustomObject with system properties
        add PSCustomObject to an array
}
Export array to profilingDatabase.csv
```

The system properties included in the database are easily extensible and should be tailored to suit the collection of any necessary data points to support your profiling needs. Any data to be included must have a common key field that identifies the simple hostname and be loaded into a PowerShell object. The provided script loops through the hosts in the targetList input file, gets properties for the computer from AD, and then uses the IndexOf method to get the property data from the modules. You can collect additional attributes from AD or import additional CSV data into objects to include in the profiling database.

### 5.3.2. createProfileDirectory.ps1

createProfileDirectory.ps1 is an interactive script that allows the user to group hosts based on system properties, filter to isolate hosts of the desired profile, and display the resulting systems. The user can then elect to proceed with establishing the profile directory. Optionally, the analysis parameter can be used to execute the analysis scripts specified in the Kansa analysis configuration file for the profile directory.

Execute the script from the Kansa output directory. During script execution, if the user elects to proceed, they are prompted for a subdirectory name for the profile directory. A directory for the profile is created in the profiles subdirectory, and symbolic links to the Kansa data collected for the hosts of the profile are created. Pseudo code for the createProfileDirectory.ps1 script is as follows:

```
Source the kansaGet-Analysis.ps1 function
Import records from profilingDatabase.csv
Display properties available for profile grouping/acquire user input
Display groups available for filtering/acquire user input
Get filtered hosts from group property
Display matching hosts/acquire user election to proceed
If elected to proceed {
        Create user specified subdirectory
        Identify Kansa module data directories
        For each directory {
                Enumerate files
                If filename exists in filtered hosts {
                        Create symbolic link to Kansa data file for host
                }
        }
        If Analysis parameter specified {
                Set function parameter variables
                Call Get-Analysis function
        }
}
```

The script leaves behind a data set that filtered to hosts matching the profile criteria. We can use any analysis methods we use with the larger Kansa data collection. The analysis option can be used just as we would during Kansa execution.

Theoretically, because we have isolated hosts that are expected to exhibit a common configuration state we expect a higher percentage of occurrences for specific data items and fewer outliers. To analyze the theory, we have to modify the Get-Analysis function to output the LogParser results to CSV format rather than TSV, making it easier to import the results to PowerShell objects. The CSV output will also be useful for discovering new unknowns relative to the baseline. We use the Get-Tasklistv.ps1 analysis script to explore the theory. The methodology for the analysis is available in the appendix. We can draw the following conclusions from the table below:

- Profiling systems based on model and organizational groupings yield more concise baselines than grouping on architecture or Windows version
- The more concise baselines have a greater number of tasks common to all hosts of the same profile
- The more concise baselines also have a greater percentage of tasks common to the group compared to total count of unique tasks

| Grouping Property | Profile Group | Hosts | Unique Tasks | Intersection | Group Unique | Common Tasks | Common/Unique |
|---|---|---|---|---|---|---|---|
| All Hosts | | 823 | 459 | | | 10 | 2.2% |
| Architecture | 32-bit | 337 | 217 | 180 | 37 | 14 | 6.5% |
| | 64-bit | 486 | 422 | 180 | 242 | 10 | 2.4% |
| Manufacturer | Dell | 735 | 390 | 42 | 242 | 12 | 3.1% |
| | Microsoft | 72 | 192 | 42 | 41 | 16 | 8.3% |
| | OEMC | 6 | 84 | 42 | 0 | 47 | 56.0% |
| | VMWare | 10 | 69 | 42 | 16 | 25 | 36.2% |
| Windows Version | Windows 7 | 740 | 407 | 95 | 254 | 11 | 2.7% |
| | Windows 8.1 | 68 | 150 | 95 | 13 | 17 | 11.3% |
| | Windows 10 | 15 | 160 | 95 | 29 | 13 | 8.1% |
| All Dell Systems | | 733 | 385 | | | 13 | |
| Model | Latitude | 61 | 254 | 170 | 80 | 29 | 11.4% |
| | OptiPlex | 672 | 308 | 170 | 135 | 18 | 5.8% |
| All Dell Optiplex7010 Win7 | | 219 | 232 | | | 17 | |
| Model/Win7/Org Unit | BackOffice | 117 | 217 | 91 | 105 | 18 | 8.3% |
| | Platforms | 46 | 119 | 91 | 9 | 24 | 20.2% |
| | Tellers | 55 | 100 | 91 | 5 | 17 | 17.0% |

We could have chosen a different module and analysis output to test the theory. A data set from a different module may reflect greater continuity across operating system versions rather than hardware model. This justifies establishing a known baseline for the data of each module independently, which does complicate the job for the analyst interested in baselining systems. It is important to be fluent in the use of the tools and have the flexibility to analyze as each specific situation dictates.

Jason Simsay, jason.simsay@gmail.com

## 5.4. Breaking it Down: Finding New Unknowns

Learning the unknown through data analysis using the provided scripts will undoubtedly take time and commitment. Not all views of the data will be useful and identifying those that are will take experimentation and experience. The process will provide a better understanding of what is normal. As with any IT/Information Security foray, beware of rabbit holes.

Once familiar with the frequency of occurrence of items from each module's data, you are ready to set the baseline. This is as simple as ensuring that the Kansa output directory is maintained for reference. Additional data collections retrieved from the environment can be compared to the baseline to identify changes within the environment. This comparison transitions our efforts from one of incident response preparation to detection. Baselines will require maintenance as the environment changes over time. However, operationalizing continuous monitoring of the current state of the environment to the known baseline is certain to yield investigations of the unknown and is an opportunity to detect an intrusion.

Security analysts comparing a current data set to a known baseline will want to acquire two pieces of intelligence. What data points are now present that were absent in the baseline? Does the new data point have a high or low frequency of occurrence? If it is high, we may have to adapt our baseline. If it is low, the next logical piece of intelligence is the identification of the host(s) that contributed the low-frequency data point.

### 5.4.1. findUnknown.ps1

Security operations teams can integrate this interactive script into routine operations to find unknown data items. The script will identify items unique to either the current or baseline collection and not present in both. The user specifies the location of the current and previously collected baseline data set. The script will look for matching profile directories within the datasets. These will have been created previously by running the createProfileDirectories.ps1 script to isolate the same profile, for each data collection. The user identifies the profile directory to analyze, and the script will locate matching analysis results files. Lastly, the user identifies the analysis results files to

compare. The compare-object cmdlet is used to identify data items unique to either the baseline or current analysis result file.

The findUnknown.ps1 script acts on multiple output directories. Execute from the Kansa directory. Unknown items or items no longer present are identified. Pseudo code is as follows:

> List output directories, prompt user for baseline and comparison directories
> Identify matching profile directories, prompt user for profile to be analyzed
> Identify matching analysis results files, prompt user to pick module analysis to compare
> Import the analysis result of interest for each data set to a PowerShell object
> Compare-object on the NoteProperty property of the PowerShell object that is not named 'CNT' and output those not equal

Afterward, it is up to the analyst to conduct additional analysis to review the systems of interest generating the outlying data item. The flexibility and capability of PowerShell make it the perfect tool for conducting further queries of the data. These queries can be constructed on the fly and are excellent for honing your skills. Example PowerShell commands for doing so are available in the appendix.

## 6. Conclusion

Establishing a rapid data collection infrastructure and knowing how to leverage it when needed is critical to efficient incident response efforts. The scalability of PowerShell Remoting makes it well suited to incident response activities, and the Kansa framework brings it home. Incidents are unique. You are not likely to be certain what to look for until it is time. Utilizing Kansa for baselining systems ahead of time and learning what is normal is akin to practice for the big game. To best handle an incident, it is important that organizations commit time and resources to establishing the necessary technology infrastructure and developing the skills of the team. You will not come away with the win when all you can do is practice, so baselining might not immediately lead you to discover anything of significance. On the other hand, it may. When something anomalous does show up in your environment, knowing what is normal will go a long way to understanding the unknown anomaly.

Jason Simsay, jason.simsay@gmail.com

Incident response programs are well documented, and your organization does not need to start from scratch. The NIST documentation is a fantastic resource. Establishing a secure PowerShell Remoting configuration in your environment will take some effort. The NSA document is the go-to source for securely enabling the powerful capabilities of Windows Remote Management (WinRM). Adversaries will be sure to find it and exploit it. Thus, it is critical to implement securely with adequate logging to detect unauthorized usage. Fortunately, PowerShell Remoting and Windows Event Log Forwarding both leverage the WS-Management protocol, specifically Microsoft's WinRM implementation. A secure implementation can support both your data collection and logging requirements. Kansa is an extensible framework developed by experts for use in live incident response scenarios. Your team must gain management support to commit the necessary resources to prepare for any incident response. Then your team must develop the needed skill to use the tools effectively when under the pressure of a real incident.

The profile baselining methods described herein are expected to help prepare your team to serve your organization. The infrastructure and tools are deployed and operational. The knowns are confirmed, and the innocuous unknowns have become familiar. Debrief your operational teams and helpdesk so they have a reference for what is normal and can better spot what is not. Operationalize the use of Kansa, analyze the data, and maintain the baselines. Discover your adversaries before someone else does.

Jason Simsay, jason.simsay@gmail.com

# References

Adams, Robert. (2015, December 7). The power and implications of enterprise incident response with PowerShell. Retrieved from https://www.sans.org/reading-room/whitepapers/incident/power-implications-enabling-powershell-remoting-enterprise-36542

Cichonski, P., Millar, T., Grance, T., & Scarfone, K. (2012, August). National Institute of Standards and Technology. Special Publication 800-61 Revision 2. Computer security incident handling guide. Retrieved from http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r2.pdf

Hosmer, Chet. (2008). WetStone Technologies, Inc. Polymorphic & metamorphic malware. Presentation at Black Hat Conference. Retrieved from https://www.blackhat.com/presentations/bh-usa-08/Hosmer/BH_US_08_Hosmer_Polymorphic_Malware.pdf

Hallenbeck, Chris. (2016, March 29). Avoiding incident response groundhog day. Retrieved from https://blog.tanium.com/avoiding-incident-response-groundhog-day/

Hull, Dave. (2014, April 14). [Web blog comment]. Retrieved from http://trustedsignal.blogspot.com/search/label/Kansa

Hull, Dave. (n.d.) Kansa. Readme.MD Retrieved from https://GitHub.com/davehull/Kansa

Hull, Dave. (2014, July 18). Kansa: A PowerShell-based incident response framework. Retrieved from http://www.powershellmagazine.com/2014/07/18/kansa-a-powershell-based-incident-response-framework/

Hull, Dave. (2011, April 23). [Web blog comment]. Retrieved from http://digital-forensics.sans.org/blog/2011/04/23/digital-forensics-least-freq-strings

M-Labs. (2012, November 7). An in-depth look into data stacking. Retrieved from https://www.fireeye.com/blog/threat-research/2012/11/indepth-data-stacking.html

Nair, Sajeev. (2013, August 7). Live response using PowerShell. Retrieved from https://www.sans.org/reading-room/whitepapers/forensics/live-response-powershell-34302

National Security Agency. (2013, December 16). Spotting the adversary with Windows event log monitoring. Retrieved from https://www.iad.gov/iad/customcf/openAttachment.cfm?FilePath=/iad/library/ia-guidance/security-configuration/applications/assets/public/upload/Spotting-the-

Adversary-with-Windows-Event-Log-Monitoring.pdf&WpKes=aF6woL7fQp3dJiwesfNwhEgT5nbAuQVRBwKBfN

Oltsik, Jon. (2015, April). Tackling attack detection and incident response. Retrieved from http://www.mcafee.com/us/resources/reports/rp-esg-tackling-attack-detection-incident-response.pdf

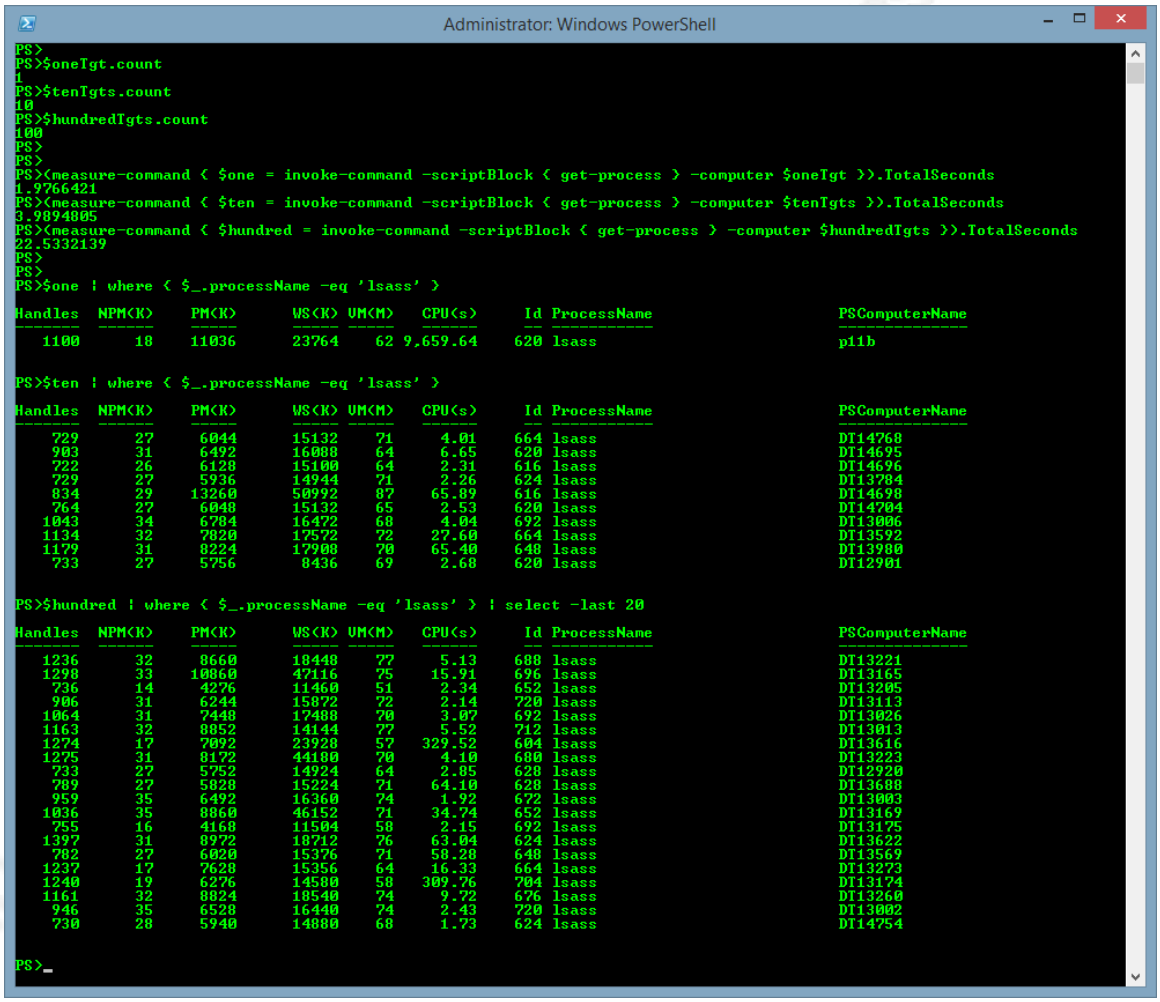
Skoudis, E., Strand, J., & SANS. (2015). SANS hacker tools, techniques, exploits & incident handling. (Vol. 1).

Tilbury, Chad. (2010, November 08). Digital forensics how-to: memory analysis with Mandiant Memoryze. Retrieved from https://digital-forensics.sans.org/blog/2010/11/08/digital-forensics-howto-memory-analysis-mandiant-memoryze/

Verizon. (2016). 2016 Data breach investigations report. Retrieved from http://www.verizonenterprise.com/resources/reports/rp_DBIR_2016_Report_en_xg.pdf

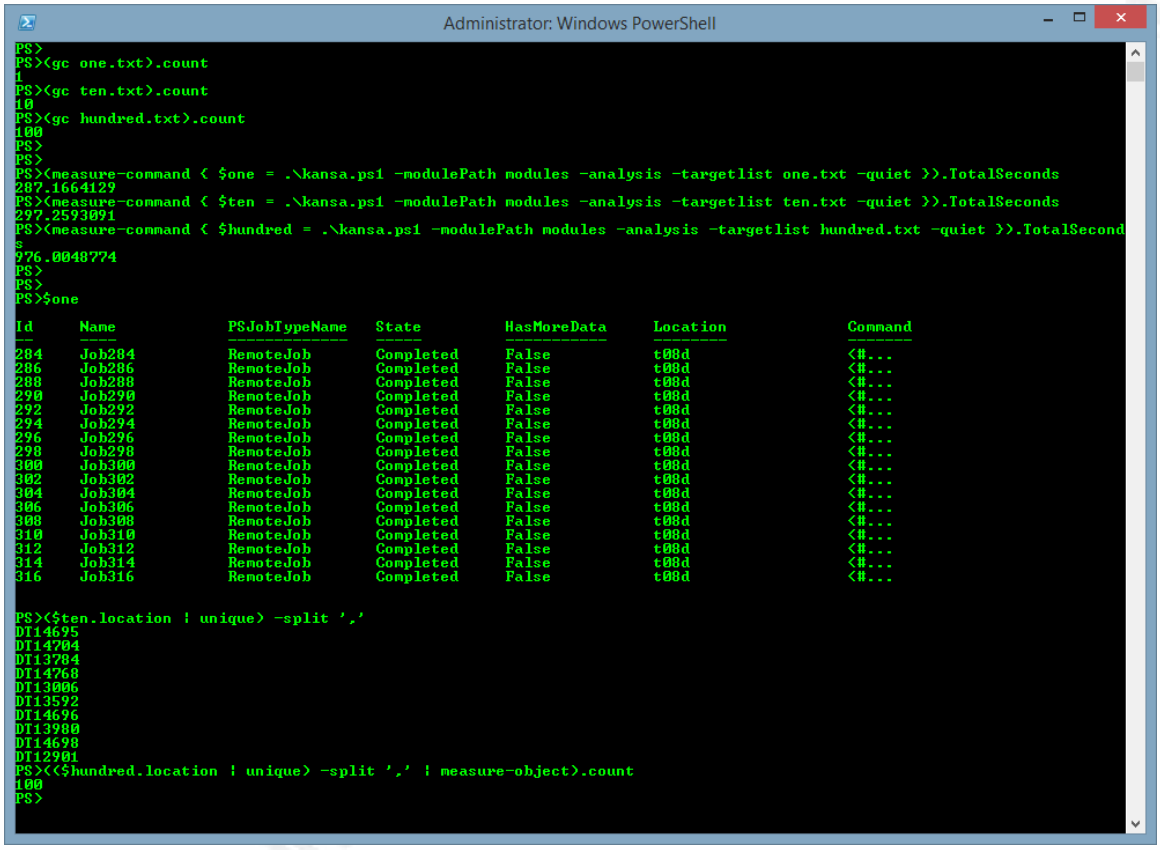
# Appendix
# Measuring Command Execution Time

The screenshot below identifies the methods and commands used to measure the invoke-command run time. We start by validating the number of hosts in each of the variables holding the target number of systems. The measure-command cmdlet is used to time the execution time for invoke-command in seconds. Finally, the output for each invoke-command is presented, depicting the expected results.




Jason Simsay, jason.simsay@gmail.com

Below we do the same as described above, except we use Kansa in place of invoke-command. Remember, we are executing 17 Kansa modules and conducting analysis whereas with invoke-command we executed a single cmdlet.




Both invoke-command and Kansa.ps1 were additionally measured when running for 861 target systems. Because there were some errors encountered, screenshots are not included here, but the data is presented in the chart. The Kansa run for 861 hosts returned upwards of 500 MB of data in about 123 minutes.

# Appendix
# Profile Analysis Methodology

The scripts that accompany this paper are available on GitHub and facilitate slicing a Kansa data collection up based on target system profile. Not all system profiles are expected to be the same in an environment and having a known baseline for the various profiles deployed in our environments is expected to be more concise and more consistent than an analysis of disparate systems. The following methodology was used to evaluate this theory.

1. Build the profile database and create profile directories for the system profiles to be analyzed. The following were chosen:
    a. Population = entire data set. Profiles include:
        i. OS Architecture
        ii. Hardware Manufacturer
        iii. Operating System
    b. Population = all Dell computers. Profiles include:
        i. Latitude
        ii. OptiPlex
    c. Population = all Dell OptiPlex computers running Windows 7. Profiles include:
        i. Back Office workstations
        ii. Platform workstations
        iii. Teller workstations
2. Record the total number of systems for each population and profile. This command, when run from the Kansa Output directory, will count the number of Tasklist data files returned by Kansa for each AnalysisResults folder, including those that have been profiled:

```
get-childitem -directory -filter Tasklistv -recurse | foreach {
    $_.fullname; get-childitem $_.fullname | measure-object | select count
}
```

3. Record the total number of unique tasklist image names for each population and profile. This command, when run from the Kansa Output directory, will count the number of unique image names found in the TaskListStack.csv analysis result:

```
get-childitem -directory -filter AnalysisReports -recurse |
get-childitem -filter TasklistStack.csv | foreach {
    $_.fullname ; $items = import-csv $_.fullname;
    $items.imagename.toupper() |
    sort | unique | measure-object | select count
}
```

4. Identify the intersection of Tasklist image names amongst the profiles.
5. Identify the Tasklist image names unique to each profile.
6. Although the above two steps were straight forward when the number of profiles was only 2, with more than 2, the various combinations of unions present complicated matters and I resorted to a spreadsheet analysis rather than script. When there are only two profiles, such as with architecture or Dell model, you can arrive at the total number of unique Tasklist images names for the population by adding the count of intersecting tasks for the profiles to the profile group unique count for each profile. For example, for the OS architecture based profiles, 180 intersecting tasks plus the 37 unique to 32-bit and the 242 unique to 64-bit yields the 459 unique tasks present in the population. The three data items below validate the accuracy of the data represented on the chart for the manufacturer, operating system, and organizational profiles.

| | |
|---|---|
| **All Count** | 42 |
| **Dell_MS_VM Count** | 2 |
| **Dell_MS_OEM Count** | 8 |
| **Dell_MS_VM Count** | 1 |
| **Dell_MS_OEM Count** | 14 |
| **Dell_MS_VM Count** | 1 |
| **Dell_MS_OEM Count** | 6 |
| **MS_VM Count** | 1 |
| **MS_OEM Count** | 11 |
| **DELL_VM Count** | 4 |
| **DELL_OEM Count** | 2 |
| **DELL_MS Count** | 55 |
| **DELL_VM Count** | 2 |
| **DELL_OEM Count** | 1 |
| **DELL_MS Count** | 10 |
| **Uniq to VM Count** | 16 |
| **Uniq to MS Count** | 41 |
| **Uniq to DELL Count** | 242 |
| **Grand Count** | 459 |

| | |
|---|---|
| **All Count** | 95 |
| **Win10 Count** | 29 |
| **Win7 Count** | 254 |
| **Win7_Win10 Count** | 26 |
| **Win7_Win8 Count** | 32 |
| **Win81 Count** | 13 |
| **Win81_Win10 Count** | 10 |

| | |
|---|---|
| == | 91 |
| Overlap BackOffice/Platform | 19 |
| Uniq to BackOffice | 105 |
| Overlap BackOffice/Teller | 3 |
| Uniq to Platform | 9 |
| Uniq To Teller | 5 |

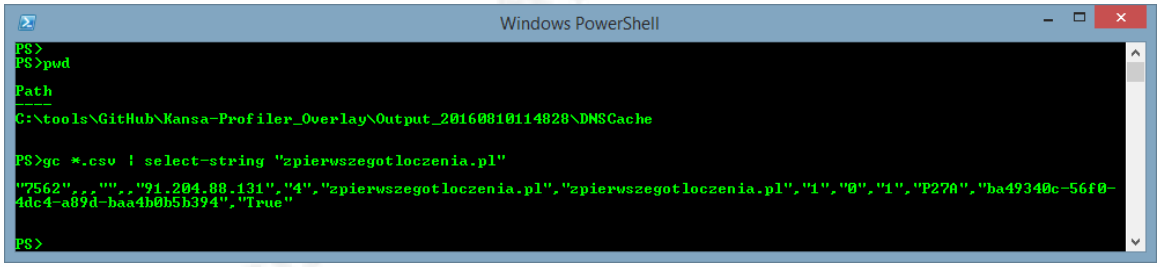
7. Identify the Tasklist image names common to the entire population and common to all hosts of the given profile.

8. The ratio of common tasks to unique tasks is a measurement of the conciseness of the baseline. A higher percentage indicates a more consistent baseline across all hosts of the profile.

# Appendix
# Next Step for the Analyst

Having executed a Kansa data collection with the analysis parameter, the analyst is left with a set of tab separated values (TSV) analysis result files. While looking at the TSV in a text editor, you discover a DNS cache entry for zpierwszegotloczenia.pl. What is the analyst's next step?

Many of the analysis scripts perform frequency analysis, and the resulting output does not identify which target host contributed this suspicious entry. To identify the host, you must return to the module data. In this case, the Get-DNSCache module outputs to the DNSCache subdirectory of the Output_ folder. From within the DNSCache directory, we can search the text of all the CSV files with this command: get-content *.csv | select-string "zpierwszegotloczenia.pl"




We have now identified the offending target host to be P27A. Next step might be to review the P27A-DNSCache.csv file to see what else is in the cache. Perhaps we must move to our web proxy logs. The analyst will need to determine the appropriate course of action based on the context of the situation.

Jason Simsay, jason.simsay@gmail.com

Last Updated: October 26th, 2020

# Upcoming SANS Training

**Click here to view a list of all SANS Courses**

| | | | |
|---|---|---|---|
| **SANS Sydney 2020** | **Sydney, AU** | **Nov 02, 2020 - Nov 14, 2020** | **Live Event** |
| **SANS Secure Thailand** | **Bangkok, TH** | **Nov 09, 2020 - Nov 14, 2020** | **Live Event** |
| **APAC ICS Summit & Training 2020** | **Singapore, SG** | **Nov 13, 2020 - Nov 28, 2020** | **Live Event** |
| **SANS Community CTF** | **,** | **Nov 19, 2020 - Nov 20, 2020** | **Self Paced** |
| **SANS Local: Oslo November 2020** | **Oslo, NO** | **Nov 23, 2020 - Nov 28, 2020** | **Live Event** |
| **SANS OnDemand** | **OnlineUS** | **Anytime** | **Self Paced** |
| **SANS SelfStudy** | **Books & MP3s OnlyUS** | **Anytime** | **Self Paced** |