CYBER DEFENSE RESEARCH & TECHNOLOGY TEAM

Certificate Authority Situational Awareness

System Design Document

3/31/2016

This System Design Document outlines the Certificate Authority Situational Awareness (CASA) Splunk application, which visualizes Certificate Authorities on network endpoints, providing ease in analysis of Certificate Authorities.

Table of Contents

Introduction

Purpose

The purpose of this document is to provide an overview of the Certificate Authorities Situational Awareness (CASA) Splunk application. The following pages provide a high level overview of the tool, design decisions, and benefits, and outline the limitations the tool possesses.

Document Overview

This document is divided into two main sections. The first section provides a general overview of the CASA components, and the second section elaborates on each component. Included in the appendices is an installation and configuration guide for installation of the CASA application, as well as a data dictionary highlighting key aspects of the data set.

Background

Previously, the process of analyzing Certificate Authorities (CAs) on the endpoints of a network was manual and time-consuming. The process required an individual to:

1) Manually scan every end-point, using the tool of choice.
2) Consolidate the results from each endpoint.
3) Compare the results against a whitelist/blacklist.
4) Correlate data between endpoints to find anomalies.
5) Remove undesirable CAs.

This process, although accurate and effective, was extremely time-consuming. As a result, the CD R&T team decided to develop a new solution that allows a user to do three things: visualize metrics on CAs across a network, compare CAs on a network against a whitelist/blacklist, and search if a given CA (based on attributes) is living anywhere on a network. This new solution could be easily deployed and provide a system administrator with nearly instant access to numerous metrics and analytics.

Vision

CASA was envisioned as a way to solve the current limitations with analyzing CA metrics on a network. The CASA Splunk application would provide simple visualizations and charts to summarize the trusted CAs found across a network, as well as searching and whitelist/blacklist comparison capabilities. Users would review the information presented in the app to look for unexpected or prohibited CAs.

High-Level Description

System Diagram

The following diagram illustrates the main components of CASA.

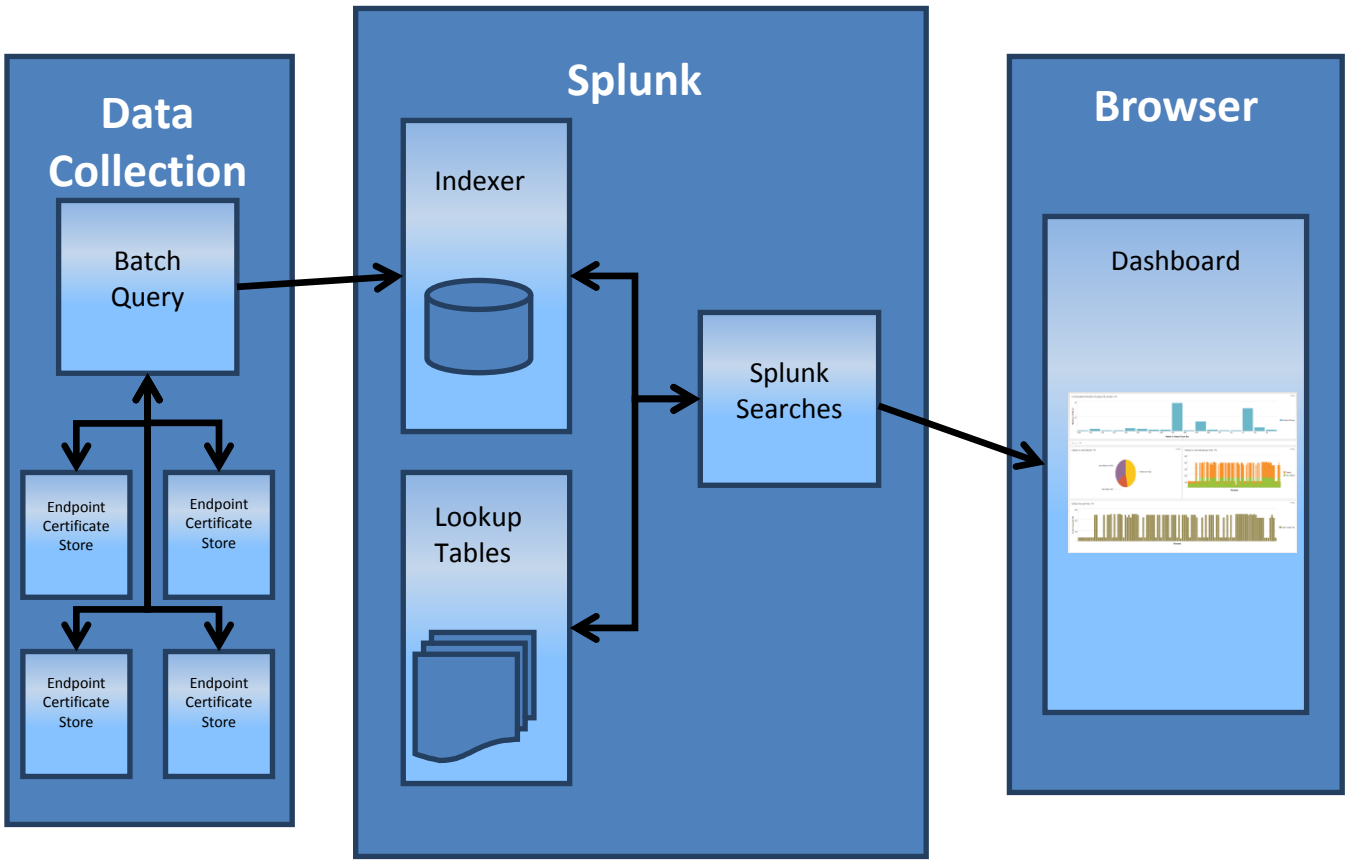


Figure 1 Abstract System Diagram

System Overview

The CASA project is divided into two pieces, working in unison: a Splunk app and a Splunk deployment app. The deployment app includes the necessary configuration files for each endpoint to collect the certificate attributes from the Windows file system send the data to the Splunk indexer. The Splunk app contains the lookups, searches, and dashboards for the user.

Data

The data for the CASA project lives in the Windows certificate store and on the Windows file system. However, the only data that is actually ingested originates from the LocalMachine store; therefore no user certificates are collected. A Batch script runs on every endpoint (or on whichever endpoints the deployment app is deployed) to collect certificate information from the Windows certificate store and sends the results to the Splunk indexer. On the Splunk indexer, in the app directory of the Windows file system, lives the lookup tables containing a list of Microsoft/Firefox installed certificates, as well as the white and black lists. These lookup tables can be altered by any consumer of the app, in order to create the desired comparisons (see Appendix A). All of this data is then used by the Splunk searches to create the visualizations and charts seen in the dashboards of the CASA app.

CAs versus Certificates

It is important to note that CASA deals with Certificate Authorities, not certificates. This means no end certificates are used in the dashboards, though they are technically collected. CASA produces metrics based on Root and Intermediate certificates, as depicted by the red box below:

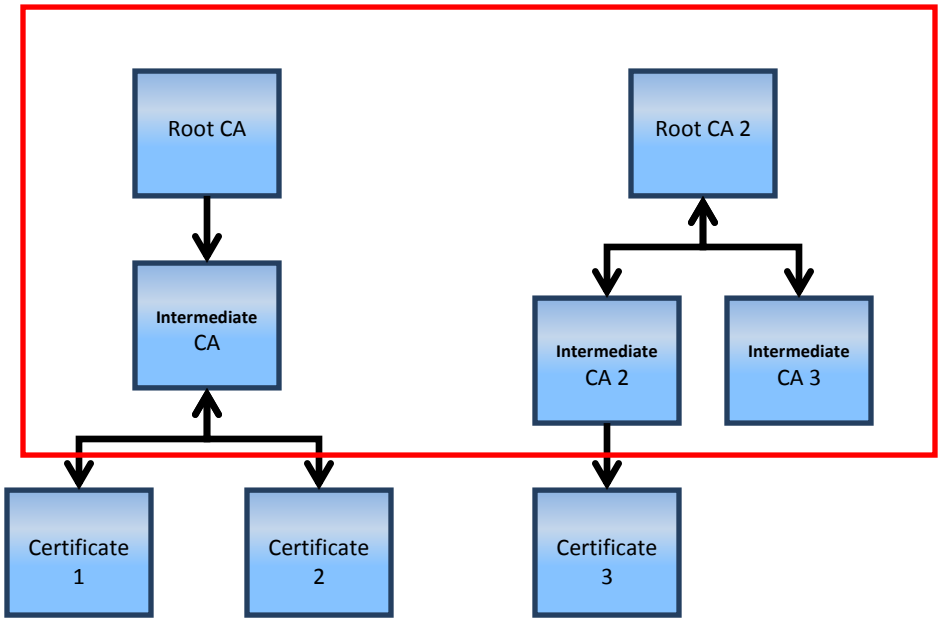


Figure 2 Certificate Trust Chain

Visualizations

There are four dashboards in the CASA app: Metrics, Analytics, CA Search, and CA Extensions.

METRICS DASHBOARD

The metrics dashboard contains numerous visualizations to assist the user in garnering some general information about the CAs across the network including: Number of CAs Per Host, Commonality of Number of Unique CAs per Host, Hosts with More than 200 CAs, Default vs. Non-Default, Default vs. Non-Default per Host, and Number of Hosts per CA.

Valid CAs per Host

This panel is a bar chart that depicts the number of trusted CAs for each host on the network. The purpose of this is to allow the user to see any boxes with an exceptionally large or small number of hosts.

Figure 3 Valid CAs per Host

Commonality of Number of Unique CAs per Host

This panel is a bar chart that depicts the number of hosts that have the displayed number of trusted CAs. This is calculated by determining the number of trusted CAs on each host then counting the number of hosts with each total. The intent of this visualization is to help a user see the distribution of CAs across the network.

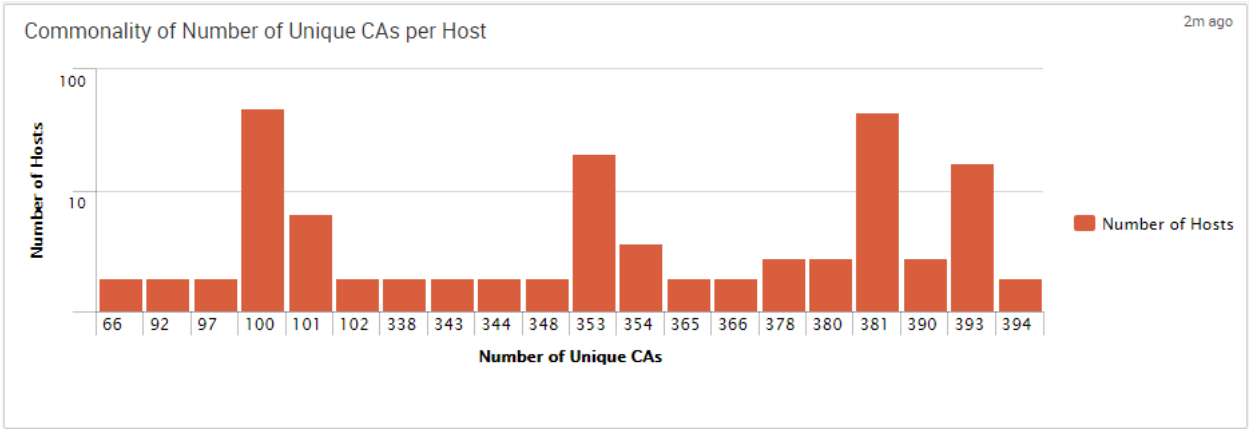


Figure 4 Commonality of Unique CAs per Host

Hosts with More than 200 CAs

This panel provides the user with a pie chart showing hosts that trust more than 200 CAs and less than 200 CAs. The number 200 was chosen based on the development network's baseline trusted CA list. The user can reconfigure this value, as necessary, to better suit their network's data.

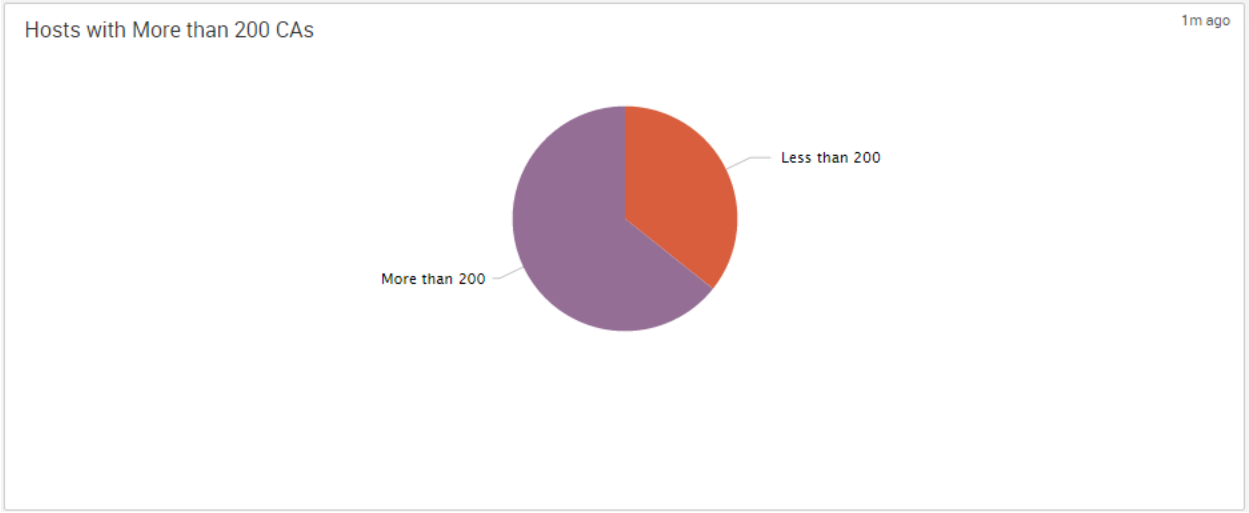


Figure 5 Hosts with More than 200 CAs

Default vs. Non-Default

This panel provides a pie chart depicting the number of unique CAs across the network that were installed by default versus those not installed by default. For the purposes of this app, default was defined by those that came as a result of installing Internet Explorer and Firefox on a host. This list can be altered by changing the lookup tables (see Appendix A). The intent of this panel is to allow a user to see any CAs that have been manually installed, or installed by other software.

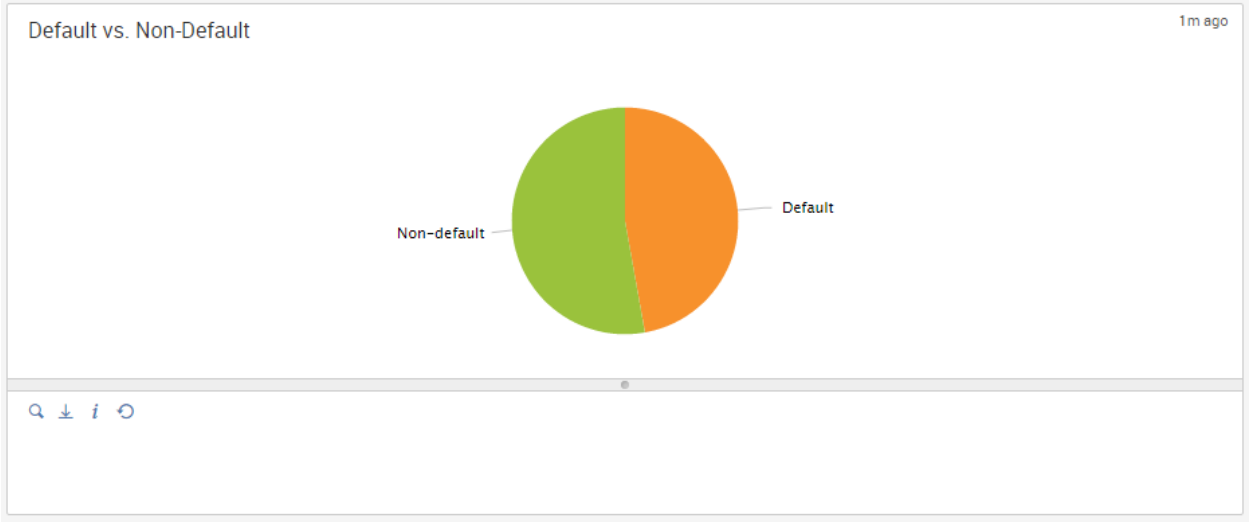


Figure 6 Default vs. Non-Default

Default vs. Non-Default per Host

This panel includes a bar chart representative of the number of certificates installed by default versus non-default, on a host by host basis. Default and non-default are defined in the same manner as above. The objective of this visualization is to show a user any box that may have a lot of CAs not installed by default.

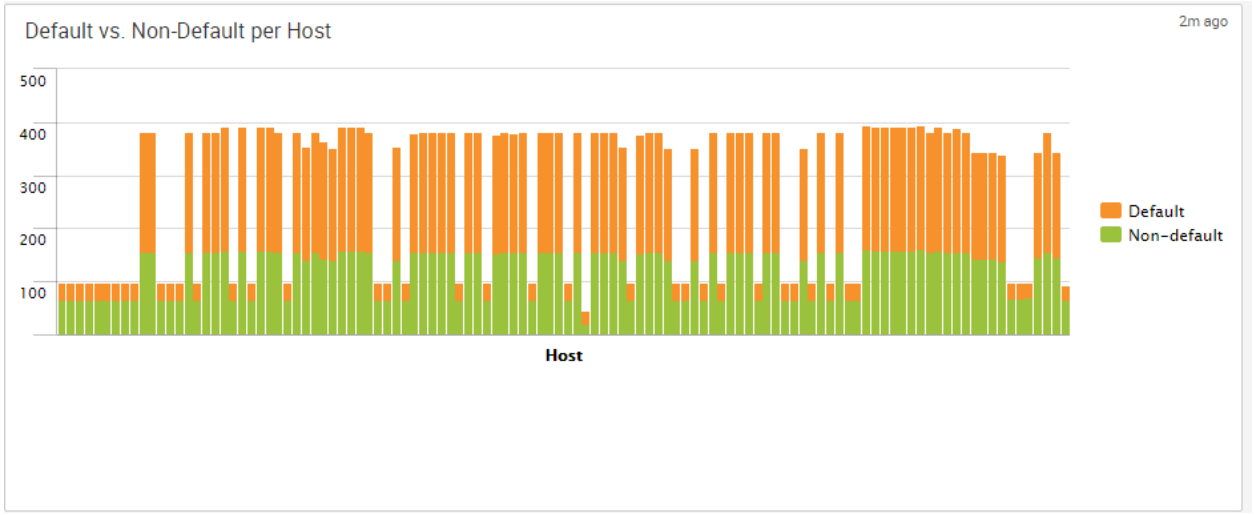


Figure 7 Default vs. Non-Default per Host

Hosts per CA

This panel displays, via a bar chart, the number of hosts that trust a given CA. The purpose of this panel is to bring to attention any CA that is trusted by only a few hosts.

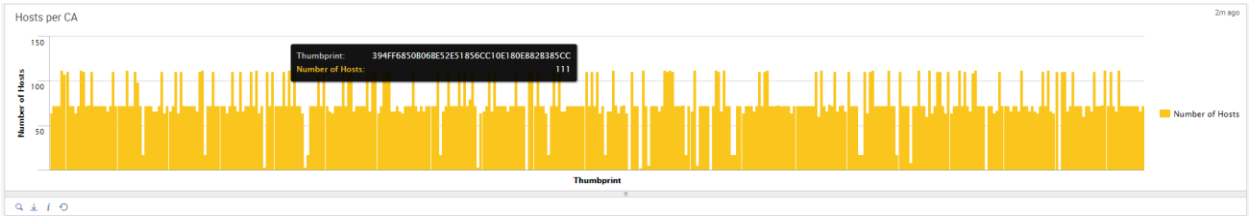


Figure 8 Number of Hosts per CA

ANALYTICS DASHBOARD

The analytics dashboard contains nine panels geared towards analytics: Time to Expiration, Validity of Intermediate CAs by Country, Country of Origin, Previously Unseen CAs, CA Algorithms, CAs Using Recommended Algorithms/Key Sizes, Invalid Trust Chain, Status of Trusted CAs, and Whitelist/Blacklist/Graylist.

Time to Expiration

This panel contains a bar graph that displays the number of CAs on a network expiring in various date ranges. The ranges depicted include: Less than 30 Days, 30-60 Days, 60-90 Days, 90-120 Days, and 120+ Days. This visualization is created by determining the time to expiration for each unique CA, then grouping the results into arbitrary time ranges.

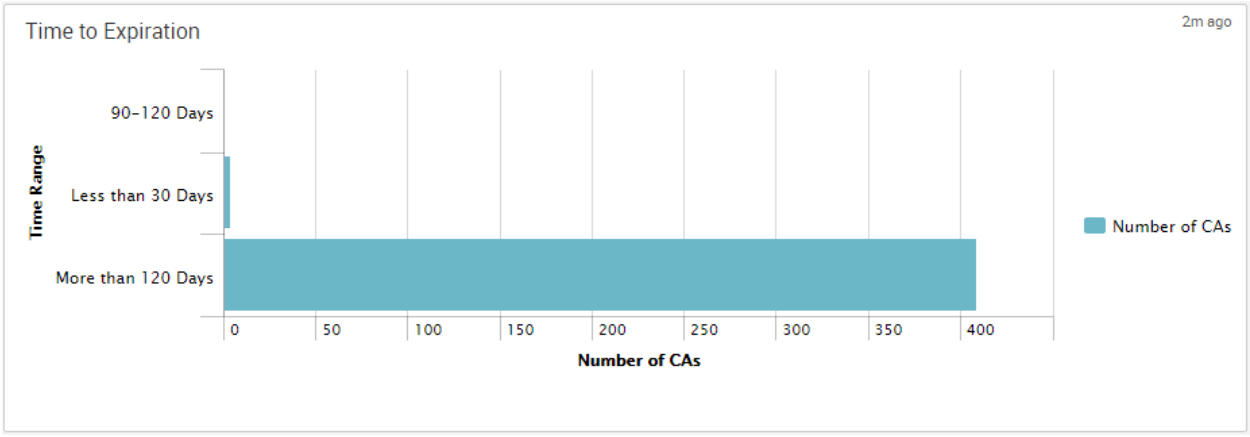


Figure 9 Time to Expiration

Validity of Intermediate CAs by Country

This panel is used to help identify potential vulnerabilities on a network, based on a previous exploitation used by hackers. Moreover, standard practice involves granting an intermediate CA a validity period of 10 years or less. This bar graph presents the number of CAs with various validity periods. Much like the time to expiration panel, this panel places CAs into bins, based on years of validity, then displays the results by country.

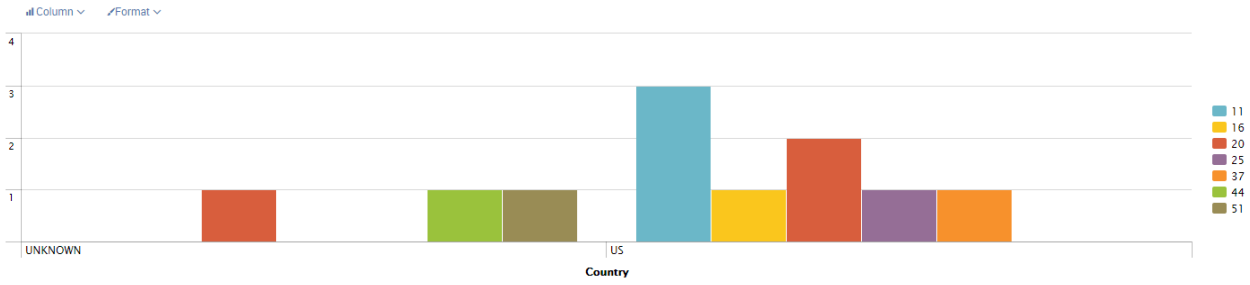


Figure 10 Intermediate CA Validity > 10 Years by Country

Country of Origin

This panel displays the number of unique CAs on a network, according to the issuing country. Once a count is obtained for the number of CAs issued by each country, the results are displayed in this pie chart.

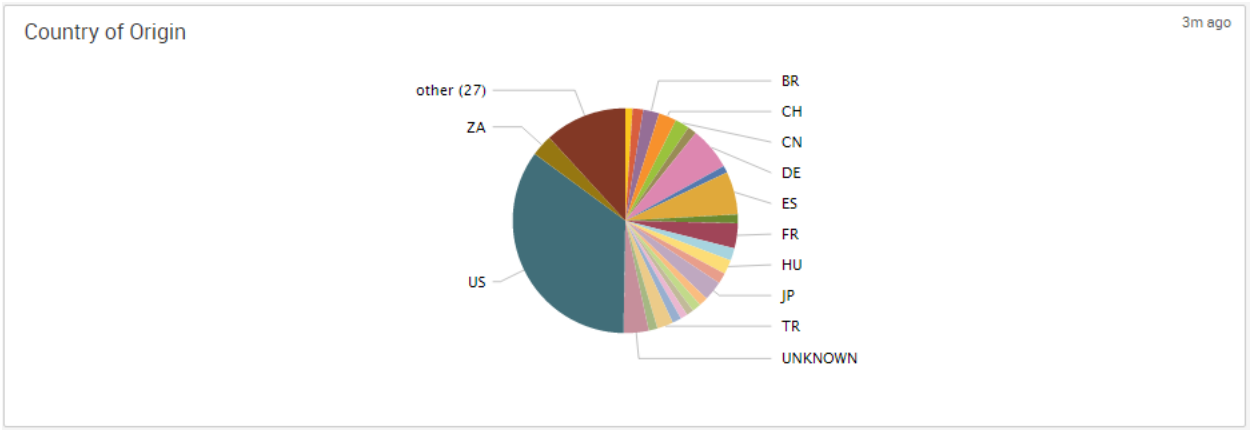


Figure 11 Country of Origin

Previously Unseen CAs

This table displays the thumbprints of certificates that have been seen for the first time within a specific timeframe. The intention here is to assist the user in discovering the times at which a CA was discovered by Splunk, possibly identifying an anomaly. The table also indicates the number of hosts that trust the CA.

20 Per Page Format Preview

Thumbprint	Number of Hosts	Days Since First Seen
E627661F5147EF3C9A065A631761828794AEB29C	1	55
FD08D102C43D5F3D0579C2DD279890FBBFB7F84D	1	55
7387A980D4A762584547BB77395F137C9D876A42	1	104
D73CA91102A2204A36459ED32213B467D7CE97FB	1	104
E490F071416EF02EDA10D5300B5317973205CAB1	1	104
888AABC5962A079B0C9B0F4C173EA0ACFD9CE795	1	106
9896387ABE170983E4A9694CE1B1130F29492A54	1	106
C9AB582BF062FF9D702C7666CA2B9AC49552628B	2	288

Figure 12 Previously Unseen CAs

CA Algorithms

This panel contains a chart which shows the count of certificates with a specific signature algorithm/public key size combination. Note, certificates using elliptic curve cryptography (i.e. ECDSA or ECDH) do not specify a key size (represented as 0 in the chart).

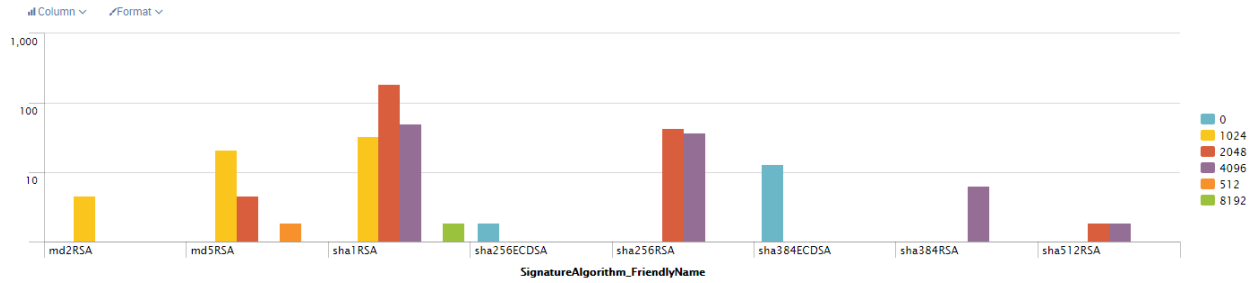


Figure 13 CA Algorithms

CAs Using Recommended Algorithms/Key Sizes

This chart groups CAs based on whether or not they are using recommended algorithms and key sizes. The recommended algorithms/key sizes used are as defined in NIST Special Publication 800-57 Part 3, Table 2-2.

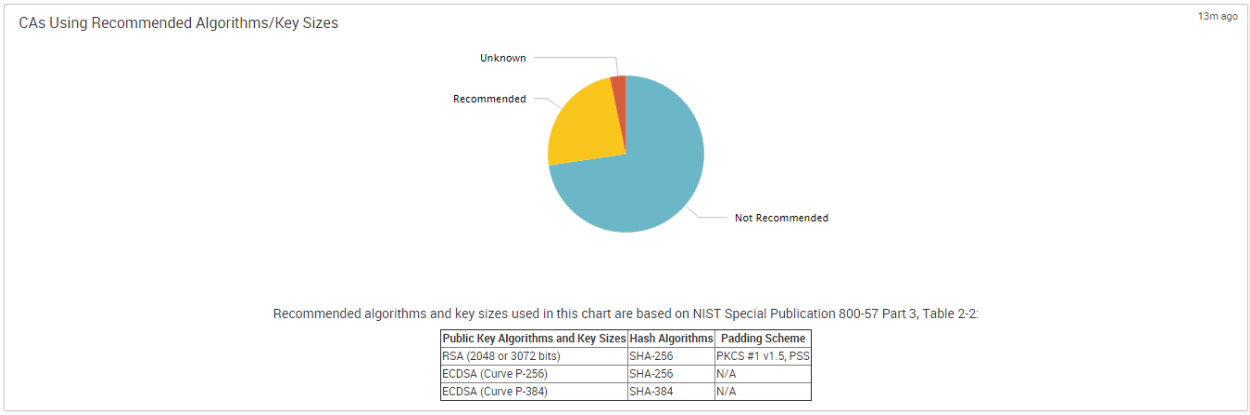


Figure 14 CAs Using Recommended Algorithms/Key Sizes

Invalid Trust Chain

This panel contains a chart with two columns: Thumbprint and Issuers. The analytic creating this chart looks for any CA that has two separate chains of trust - or two separate Issuers (see Appendix B for information on trust chains). Any CAs contained in this chart should be investigated further, as it could be a sign of malicious behavior.

Status of Trusted CAs

This panel utilizes the whitelist and blacklist in the lookup tables to determine whether each CA is part of a whitelist, blacklist, or graylist (status unknown). For every unique CA on a network, the Thumbprint is compared to a list of Thumbprints on a whitelist and blacklist to determine which, if either, the CA belongs to. The CAs are placed into bins, according to their status, and then counted to determine the number of white, black, and gray for display in this bar graph.

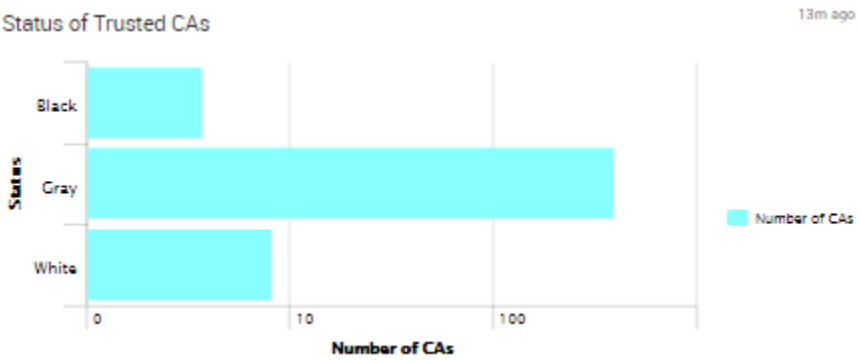


Figure 15 Status of Trusted CAs

Whitelist/Blacklist/Graylist

This chart provides the status of each CA on a network, after comparing them against the whitelist and blacklist in the manner previously mentioned.

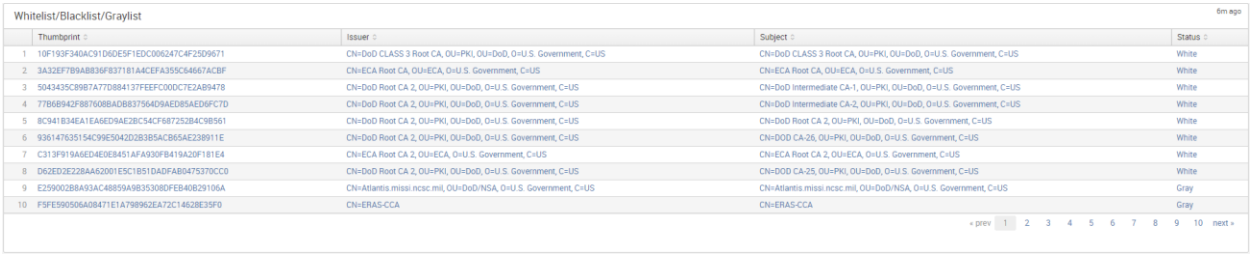


Figure 16 Whitelist/Blacklist/Graylist

CA Search

The third dashboard contains six inputs and a table. The inputs are Time Range, Host, Issuer, Thumbprint, Signature Algorithm, and Key Size. The idea here is that the user will enter a value for any of the six fields to pinpoint where, if at all, a CA exists on the network. Moreover, once the inputs have been selected, the table included in the panel will update accordingly. The goal is to help a user search for a CA based on the Host, Issuer, Thumbprint, Signature Algorithm, and/or Key Size.

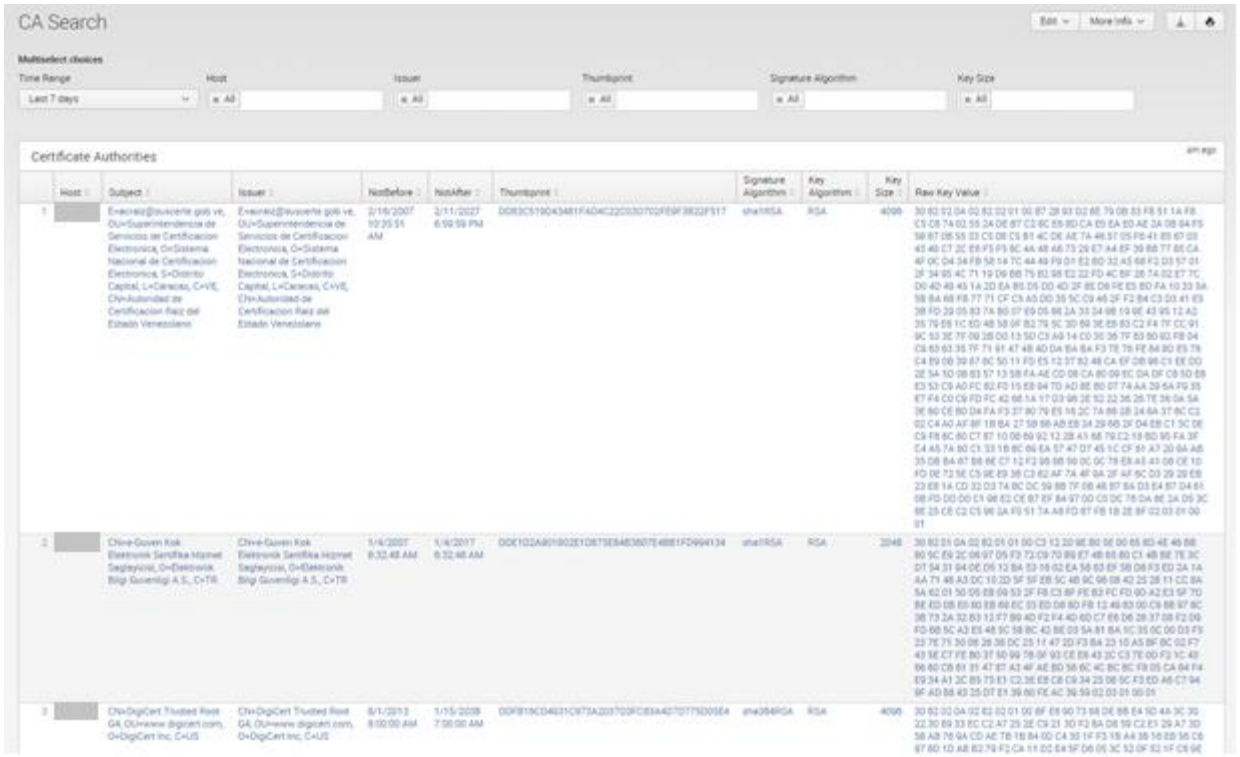
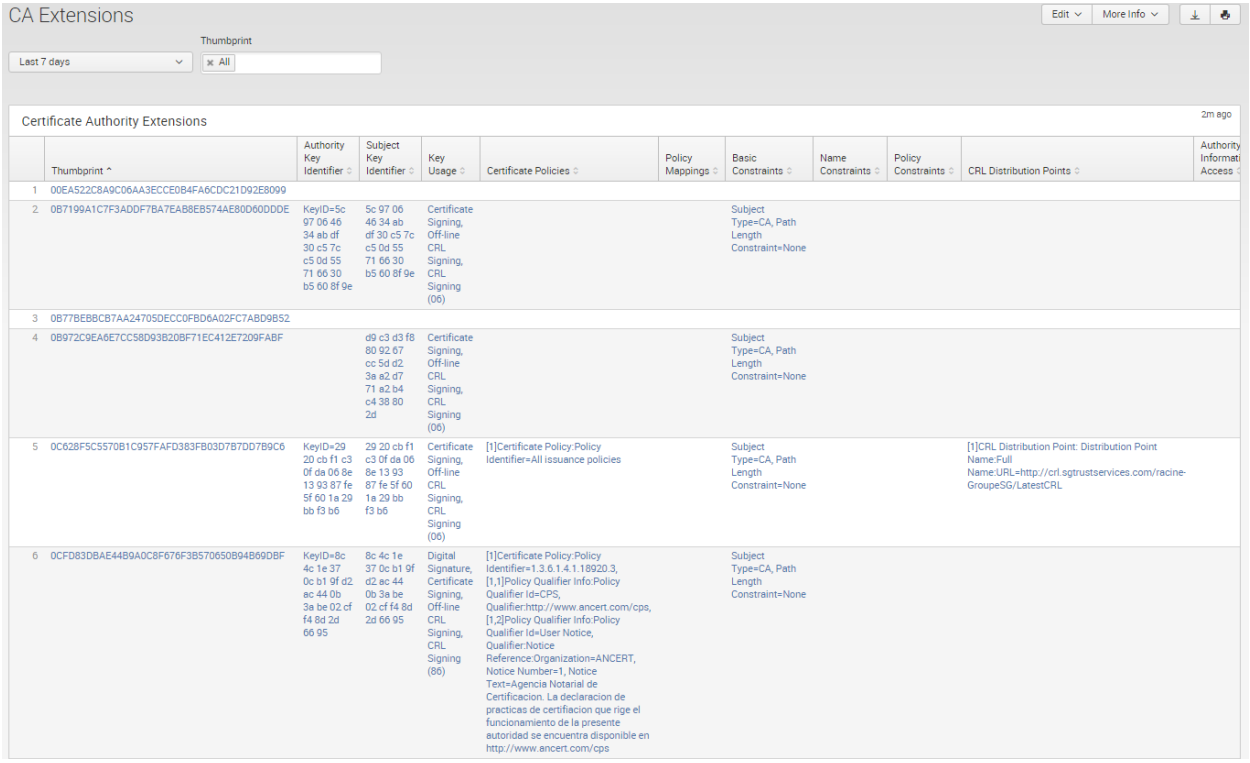


Figure 17 CA Search

CA Extensions

The fourth and final dashboard contains two inputs (Time Range and Thumbprint) and a table, which displays the certificate extensions for the selected thumbprint(s). The intent is to provide users insight into which extensions are used by each CA, and the corresponding extension information.



	Thumbprint ^	Authority Key Identifier	Subject Key Identifier	Key Usage	Certificate Policies	Policy Mappings	Basic Constraints	Name Constraints	Policy Constraints	CRL Distribution Points	Authority Informati Access
1	00EA522C8A9C06AA3ECCE0B4FA6CDC21D92E8099										
2	0B7199A1C7F3ADDF7BA7EAB8EB574AE80D60DDDE	KeyID=5c 97 06 46 34 ab df 30 c5 7c c5 0d 55 71 66 30 b5 60 8f 9e	5c 97 06 46 34 ab df 30 c5 7c c5 0d 55 71 66 30 b5 60 8f 9e	Certificate Signing, Off-line CRL Signing, CRL Signing (06)			Subject Type=CA, Path Length Constraint=None				
3	0B77BEBBCB7AA24705DECC0FBD6A02FC7ABD9B52										
4	0B972C9EA6E7CC58D93B20BF71EC412E7209FABF		d9 c3 d3 f8 80 92 67 cc 5d d2 3a a2 d7 71 a2 b4 c4 38 80 2d	Certificate Signing, Off-line CRL Signing, CRL Signing (06)			Subject Type=CA, Path Length Constraint=None				
5	0C628F5C5570B1C957FAFD383FB03D7B7DD7B9C6	KeyID=29 20 cb f1 c3 0f da 06 8e 13 93 87 fe 5f 60 1a 29 bb f3 b6	29 20 cb f1 c3 0f da 06 8e 13 93 87 fe 5f 60 1a 29 bb f3 b6	Certificate Signing, Off-line CRL Signing, CRL Signing (06)	[1]Certificate Policy:Policy Identifier=All issuance policies		Subject Type=CA, Path Length Constraint=None			[1]CRL Distribution Point: Distribution Point Name:Full Name:URL=http://crl.sgtrustservices.com/racine-GroupeSG/LatestCRL	
6	0CFD83DBAE44B9A0C8F676F3B570650B94B69DBF	KeyID=8c 4c 1e 37 0c b1 9f d2 ac 44 0b 3a be 02 cf f4 8d 2d 66 95	8c 4c 1e 37 0c b1 9f d2 ac 44 0b 3a be 02 cf f4 8d 2d 66 95	Digital Signature, Certificate Signing, Off-line CRL Signing, CRL Signing (86)	[1]Certificate Policy:Policy Identifier=1.3.6.1.4.1.18920.3, [1,1]Policy Qualifier Info:Policy Qualifier Id=CPS, Qualifier:http://www.ancert.com/cps, [1,2]Policy Qualifier Info:Policy Qualifier Id=User Notice, Qualifier:Notice Reference:Organization=ANCERT, Notice Number=1, Notice Text=Agencia Notarial de Certificacion. La declaracion de practicas de certifiacion que rige el funcionamiento de la presente autoridad se encuentra disponible en http://www.ancert.com/cps		Subject Type=CA, Path Length Constraint=None				

Figure 18 CA Extensions

Limitations

Currently, the CASA app has not been extensively tested. Some of these limitations are explained in the following sections.

LocalMachine versus *CurrentUser* Store

As previously mentioned, the current Batch scripts pull data only from the LocalMachine certificate store. The CurrentUser certificate store is only generated when a user logs in, and as there is no guarantee that a user will be logged in when the Batch script is run, using this store may result in an overall picture that is not entirely accurate. This means that a user may have custom CAs installed that will be noticed by the Batch script. Though this was a design decision, it is still a limitation.

Appendix A

Installation and Configuration Guide

The following is a guide to assist with the installation of CASA into Splunk, as well as with the configuration of its customizable components.

Installation Prerequisites

Software Requirements

1. Windows Operating System
2. Splunk 6 or higher

Permission Requirements

Splunk must have permission to execute PowerShell commands.

Installation

The Splunk app should be installed prior to any ingestion of data (ie, installing the deployment app).

CASA Splunk App

Create an index called cert_auth in order to minimize alteration of the searches. The CASA Splunk App should not require any additional configuration and can simply be placed within the directory **$SPLUNK_HOME\etc\apps**. The hosting Splunk server(s) needs to be restarted to activate the app.

CASA Splunk Deployment App

The CASA Splunk Deployment App found at this location $SPLUNK_HOME\etc\apps\casa\casa should be moved to the deployment-apps directory on the Deployment Server -$SPLUNK_HOME\etc\deployment-apps. The app should then be added, via the GUI, to the desired server class. If a deployment server is not used, the app should be unzipped and placed in the apps directory on the desired Universal Forwarders (**$SPLUNK_HOME\etc\apps)**.

Configuration

Both the CASA deployment and CASA Splunk apps should work automatically without configuration. However, there are a few user preferences that can be modified relatively easily.

Altering the Lookup Tables

New Microsoft XML GPO files can easily be added to the SPOC Technology Add-on app. Within the **$SPLUNK_HOME\etc\apps\casa\lookups** folder, there exists three lookup table files: DefaultCAList.csv, Whitelist.csv, and Blacklist.csv. To alter any of these lists, open the file in a text editor and save the changes; the lookup tables will be automatically updated.

Altering the Frequency of Collection

The current configuration allows for collection of CAs once a week. To alter this collection two changes must be made.

- The first change is in the deployment app. Navigate to **$SPLUNK_HOME\etc\deployment-apps\casa\local** and open inputs.conf. Alter the cron schedule to the desired range, save the changes, and redeploy the app to the desired hosts (Note: the changes must be made on all of the hosts).
- The second change is in the searches within the CASA app. Navigate to **$SPLUNK_HOME\etc\apps\casa\local** and open savedsearches.conf. For every saved search, find the *search* key and alter the *hoursago* argument to the desired ranged. Preferably, this integer would correlate to the time lapse between ingest cycles. Restart Splunk.

Appendix B

Data Dictionary

The following information is intended to assist in the understanding of the data ingested into Splunk.

Data Fields

The following is a list of the standard x.509 fields and their meaning, as well as the correlation to the field names returned by the Batch script.

X.509 Attribute	PowerShell Attribute	Description
Version	Version	Version of the certificate
Serial Number	SerialNumber	Serial number of the certificate
Issuer	Issuer	The issuer of the certificate
Not Before	NotBefore	The date the certificate becomes valid
Not After	NotAfter	The date the certificate becomes invalid
Subject	Subject	The subject of the certificate
Thumbprint	Thumbprint	The unique thumbprint of the certificate

Figure 19 Data Attributes

Trust Chains

To have a valid trust chain, an intermediate CA's *Issuer* value must match the *Subject* value of its parent. Below is an example

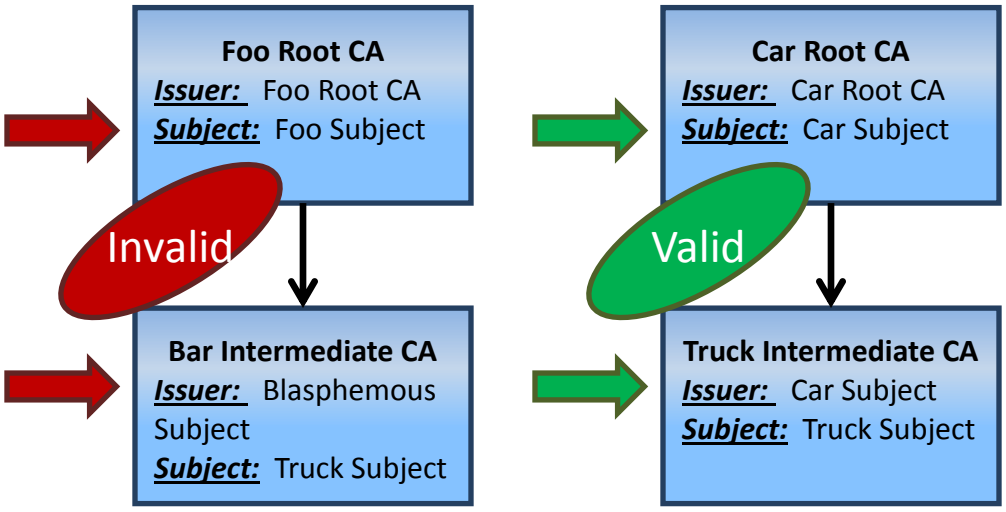


Figure 20 Valid vs. Invalid Trust Chains

Understanding Store Names

Certificates belong to various logical data stores, based on whether they have the ability to issue other certificates (CA) or not. For example, an Intermediate CA belongs in one store, a Root CA belongs in another, and a user certificate belongs in another. The CASA app benefits from this information, as it makes it simple to distinguish between certificates and Certificate Authorities. Although this

information is not contained within the certificates' attributes, the Batch script PowerShell query provides this information for us in a field called PSParentPath.

All values of the PSParentPath field begin with *Microsoft.PowerShell.Security\Certificate::LocalMachine* followed by the store name. As such, CASA removes this portion of the field value, leaving only the store name. Since the CASA app is only interested in CAs, the searches are filtered based on this remaining portion, leaving only those stores that contain CAs (Root, CA, AuthRoot).

Issuing Country

Both the Subject and Issuer fields follow the same structure, which contains numerous attribute-value pairs. An example of these fields, as well as an explanation of each attribute, can be viewed in the table below. The most common of these attributes can be seen in the following table:

CN=Sample Cert, OU=R&D, O=Company Inc., L=Baltimore, S=Maryland, C=US		
Attribute	**Full Name**	**Explanation**
CN	CommonName	This is a common name given to a certificate
OU	OrganizationalUnit	The unit, within an organization, that issued the certificate
O	Organization	The organization that issued the certificate
L	Locality	The city, or locality, from which the issuing organization is headquarted
S	State	The state/province in which the city is located
C	Country	The country to which the state belongs

Figure 21 Issuer Field Example

Based on the information presented in this table, the example certificate would be called Sample Cert. The Sample Cert certificate was issued by the R&D Unit of Company Inc., which is headquartered in Baltimore, MD, US. The CASA app makes use of this data structure by parsing the Country field from each CA, for use in metrics calculations.

Using Thumbprints

Throughout the CASA app, the Thumbprint field is used exclusively as the way to differentiate and dedup the various CAs. The Thumbprint field is a hash that is unique to each and every CA. As such, it was determined to be the best field to distinguish CAs from one another.

Data Query

CASA makes use of three tags to capture the desired events and allow for easy configuration in different environments: certificate_index, certificate_source, and certificate_authority. The default values used for each tag are defined below.

- certificate_index: index=cert_auth
- certificate_source: source=*certificateList.csv
- certificate_authority: PSParentPath=Microsoft.PowerShell.Security\Certificate::LocalMachine\AuthRoot, PSParentPath=Microsoft.PowerShell.Security\Certificate::LocalMachine\CA, PSParentPath=Microsoft.PowerShell.Security\Certificate::LocalMachine\Root

The tags and additional terms make up the following root search, from which all visualization searches are based:

tag=certificate_index tag=certificate_source tag=certificate_authority hoursago=168 | eval ed2=strptime(NotAfter, "%m/%d/%Y %I:%M:%S %p") | eval now=now() | eval timeDiff=ed2-now | where timeDiff>0

The table below explains what each part of this search does, and the purpose for doing it:

Search Syntax	Result	Purpose
tag=certificate_index tag=certificate_source tag=certificate_authority hoursago=168	Returns most recent, full data set, limiting the results to only Certificate Authorities	Gets the data
eval ed2=strptime(NotAfter, "%m/%d/%Y %I:%M:%S %p")	Strips the time from the expiration field	Determines the expiration date
eval now=now() | eval timeDiff=ed2-now | where timeDiff>0	Compares the time to now, to determine if the certificate is still valid and trusted	Filters to only valid, trusted CAs

Figure 21 Root Query Explained

In addition to this root search, each visualization is backed by additional syntax in its search. For the most part, these involve gathering distinct counts based on various criteria. The next section provides the searches for each visualization.

Searches

The following are the Splunk searches for the corresponding visualization.

Valid CAs per Host

- Uniquely count the number of CAs per host: *dc(Thumbprint) by host*

REGULAR SPL

tag=certificate_index tag=certificate_source tag=certificate_authority hoursago=168 | eval ed2=strptime(ssl_end_time, "%m/%d/%Y %I:%M:%S %p") | eval now=now() | eval timeDiff=ed2-now | where timeDiff>0 | stats dc(Thumbprint) by host | rename dc(Thumbprint) AS "Number of CAs" host AS "Host"

PIVOT SPL

| pivot Certificate_Authorities Certificate_Authority_Stores dc(Thumbprint) AS "Distinct Count of Thumbprint" SPLITROW host AS Host FILTER Expired = false SORT 10000 host ROWSUMMARY 0 COLSUMMARY 0 NUMCOLS 0 SHOWOTHER 1

Hosts with More than 200 CAs

1. Uniquely count the number of CAs per host: *dc(Thumbprint) AS dcount by host*
2. Bin the results based on whether the count is more or less than 200: *eval more=if(dcount>200,"More than 200","Less than 200") | stats count by more*

REGULAR SPL

tag=certificate_index tag=certificate_source tag=certificate_authority hoursago=168 | eval ed2=strptime(ssl_end_time, "%m/%d/%Y %I:%M:%S

%p") | eval now=now() | eval timeDiff=ed2-now | where timeDiff>0 | stats dc(Thumbprint) AS dcount by host | eval more=if(dcount>200,"More than 200","Less than 200") | stats count by more | rename count AS "Total Number of Hosts" more AS "Number of Unique CAs"

DATA MODEL DRIVEN SPL

| tstats dc(X509_Certificate_Standard.Thumbprint) AS dcount FROM datamodel=Certificate_Authorities where earliest=-7d (X509_Certificate_Standard.StoreName="CA" OR X509_Certificate_Standard.StoreName="Root" OR X509_Certificate_Standard.StoreName="AuthRoot") X509_Certificate_Standard.Certificate_Authority_Stores.Expired=False by host | eval more=if(dcount>200,"More than 200","Less than 200") | stats count by more | rename count AS "Total Number of Hosts" more AS "Number of CAs"

Default vs. Non-Default per Host

1. Using the DefaultCAList.csv lookup file, determine if the CA has a browser identified and if so mark as "Default", else mark as "Non-default": *lookup DefaultCAList.csv "SHA-1 Fingerprint" AS Thumbprint | eval n=if(isnull(Browser),"Non-default","Default")*
2. Plot the count of thumbprints as a stacked column chart, per host and default category: *chart dc(Thumbprint) by host, n*

REGULAR SPL

tag=certificate_index tag=certificate_source tag=certificate_authority hoursago=168 | lookup DefaultCAList.csv "SHA-1 Fingerprint" AS Thumbprint | eval ed2=strptime(ssl_end_time, "%m/%d/%Y %I:%M:%S %p") | eval now=now() | eval timeDiff=ed2-now | where timeDiff>0 | eval n=if(isnull(Browser),"Non-default","Default") | chart dc(Thumbprint) by host, n | rename dc(Thumbprint) AS "Number of CAs" host AS "Host"

DATA MODEL DRIVEN SPL

| tstats dc(X509_Certificate_Standard.Thumbprint) FROM datamodel=Certificate_Authorities where earliest=-7d (X509_Certificate_Standard.StoreName="CA" OR X509_Certificate_Standard.StoreName="Root" OR X509_Certificate_Standard.StoreName="AuthRoot") by X509_Certificate_Standard.Certificate_Authority_Stores.Default_Status | rename dc(X509_Certificate_Standard.Thumbprint) AS "Number of CAs" X509_Certificate_Standard.Certificate_Authority_Stores.Default_Status AS Status

Invalid Trust Chain

1. For each thumbprint, determine issuer(s): *chart values(ssl_issuer) AS "Issuers" by Thumbprint*
2. Limit to invalid CAs (more than one issuer): *where mvcount(Issuers)>1*

REGULAR SPL

tag=certificate_index tag=certificate_source tag=certificate_authority hoursago=168 | chart values(ssl_issuer) AS "Issuers" by Thumbprint | where mvcount(Issuers)>1 | mvexpand Issuers

DATA MODEL DRIVEN SPL

| tstats values(X509_Certificate_Standard.Issuer) AS Issuers FROM datamodel=Certificate_Authorities where earliest=-7d (X509_Certificate_Standard.StoreName="CA" OR X509_Certificate_Standard.StoreName="Root" OR X509_Certificate_Standard.StoreName="AuthRoot") by X509_Certificate_Standard.Thumbprint | rename X509_Certificate_Standard.Thumbprint AS Thumbprint | where mvcount(Issuers)>1 | mvexpand Issuers

Status of Trusted CAs

- Using the Blacklist.csv and Whitelist.csv lookups, determine whether the CA is on the blacklist, whitelist, both, or neither ("gray")
    a. White if on whitelist and not on blacklist: *isnotnull(white) AND isnull(black), "White"*
    b. Black if not on whitelist and on blacklist: *isnull(white) AND isnotnull(black), "Black"*
    c. Unknown ("gray") if on neither whitelist nor blacklist: *isnull(white) AND isnull(black), "Gray"*
    d. Both if on whitelist and on blacklist: *isnotnull(white) AND isnotnull(black), "Appears in Whitelist and Blacklist"*

REGULAR SPL

tag=certificate_index tag=certificate_source tag=certificate_authority hoursago=168 | eval ed2=strptime(ssl_end_time, "%m/%d/%Y %I:%M:%S %p") | eval now=now() | eval timeDiff=ed2-now | where timeDiff>0 | lookup Whitelist.csv Good_Thumbprint AS Thumbprint OUTPUTNEW Good_Thumbprint AS white | lookup Blacklist.csv Bad_Thumbprint AS Thumbprint OUTPUTNEW Bad_Thumbprint AS black | eval Status=case(isnotnull(white) AND isnull(black), "White", isnull(white) AND isnotnull(black), "Black", isnull(white) AND isnull(black), "Gray", isnotnull(white) AND isnotnull(black), "Appears in Whitelist and Blacklist") | stats dc(Thumbprint) AS "Number of CAs" by Status

DATA MODEL DRIVEN SPL

| tstats dc(X509_Certificate_Standard.Thumbprint) FROM datamodel=Certificate_Authorities where earliest=-7d (X509_Certificate_Standard.StoreName="CA" OR X509_Certificate_Standard.StoreName="Root" OR X509_Certificate_Standard.StoreName="AuthRoot") X509_Certificate_Standard.Certificate_Authority_Stores.Expired=False by X509_Certificate_Standard.Certificate_Authority_Stores.White_Black_List_Status | rename dc(X509_Certificate_Standard.Thumbprint) AS "Number of CAs" X509_Certificate_Standard.Certificate_Authority_Stores.White_Black_List_Status AS Status

Country of Origin

1. Determine issuing country from Issuer field: rex field=ssl_issuer "C=(?<Country>[a-zA-Z]*)"
2. If no country specified, mark as UNKNOWN: eval Country=coalesce(upper(Country), "UNKNOWN")

REGULAR SPL

tag=certificate_index tag=certificate_source tag=certificate_authority hoursago=168 | eval ed2=strptime(ssl_end_time, "%m/%d/%Y %I:%M:%S %p") | eval now=now() | eval timeDiff=ed2-now | where timeDiff>0 | rex field=ssl_issuer "C=(?<Country>[a-zA-Z]*)" | eval Country=coalesce(upper(Country), "UNKNOWN") | stats dc(Thumbprint) by Country | rename dc(Thumbprint) AS "Number of CAs"

DATA MODEL DRIVEN

| tstats dc(X509_Certificate_Standard.Thumbprint) FROM datamodel=Certificate_Authorities where earliest=-7d (X509_Certificate_Standard.StoreName="CA" OR X509_Certificate_Standard.StoreName="Root" OR X509_Certificate_Standard.StoreName="AuthRoot") X509_Certificate_Standard.Certificate_Authority_Stores.Expired=False by X509_Certificate_Standard.Certificate_Authority_Stores.Normalized_Issuing_Country | rename X509_Certificate_Standard.Certificate_Authority_Stores.Normalized_Issuing_Country AS "Issuing Country" dc(X509_Certificate_Standard.Thumbprint) AS "Number of CAs"

Validity of Intermediate CAs by Country

1. Determine validity period (time between NotBefore and NotAfter), limit to CAs with validity > 10 years: *eval ed1=strptime(NotBefore, "%m/%d/%Y %I:%M:%S %p") | eval ed2=strptime(NotAfter, "%m/%d/%Y %I:%M:%S %p") | eval now=now() | eval timeDiff=ed2-now | eval period=ed2-ed1 | eval yearsOfValidity=ceil(period/31536000) | where timeDiff>0 AND yearsOfValidity>10*
2. Determine country, normalize: *rex field=ssl_issuer "C=(?<Country>[a-zA-Z]*)" | eval Country=coalesce(upper(Country), "UNKNOWN")*

REGULAR SPL

tag=certificate_index tag=certificate_source hoursago=168 | where PSParentPath="Microsoft.PowerShell.Security\Certificate::LocalMachine\CA" | eval ed1=strptime(ssl_start_time, "%m/%d/%Y %I:%M:%S %p") | eval ed2=strptime(ssl_end_time, "%m/%d/%Y %I:%M:%S %p") | eval now=now() | eval timeDiff=ed2-now | eval period=ed2-ed1 | eval yearsOfValidity=ceil(period/31536000) | where timeDiff>0 AND yearsOfValidity>10 | rex field=ssl_issuer "C=(?<Country>[a-zA-Z]*)" | eval Country=coalesce(upper(Country), "UNKNOWN") | chart dc(Thumbprint) by Country, yearsOfValidity | rename dc(Thumbprint) AS "Number of CAs" yearsOfValidity AS "Years of Validity" ssl_start_time AS "Date of Validity" ssl_end_time AS "Date of Expiration"

DATA MODEL DRIVEN SPL

| tstats dc(X509_Certificate_Standard.Thumbprint) FROM datamodel=Certificate_Authorities where earliest=-7d (X509_Certificate_Standard.StoreName="CA" OR X509_Certificate_Standard.StoreName="Root" OR X509_Certificate_Standard.StoreName="AuthRoot") X509_Certificate_Standard.Certificate_Authority_Stores.Expired=False by X509_Certificate_Standard.Certificate_Authority_Stores.Normalized_Issuing_Country | rename X509_Certificate_Standard.Certificate_Authority_Stores.Normalized_Issuing_Country AS "Issuing Country" dc(X509_Certificate_Standard.Thumbprint) AS "Number of CAs"

Time to Expiration

1. Determine number of days between current day and expiration date: *eval ed2=strptime(NotAfter, "%m/%d/%Y %I:%M:%S %p") | eval now=now() | eval timeDiff=ed2-now | where timeDiff>0 | eval daysToExpire=timeDiff/86400*
2. Bin CAs based on days until expiration: *eval timeRange=case(daysToExpire>=120, "More than 120 Days", daysToExpire<120 AND daysToExpire>=90, "90-120 Days", daysToExpire<90 AND daysToExpire>=60, "60-90 Days", daysToExpire<60 AND daysToExpire>=30, "30-60 Days", daysToExpire<30 AND daysToExpire>0, "Less than 30 Days", daysToExpire<=0, "Expired")*

REGULAR SPL

tag=certificate_index tag=certificate_source tag=certificate_authority hoursago=168 | eval daysToExpire=(strptime(ssl_end_time, "%m/%d/%Y %I:%M:%S %p") - now()) / 86400 | eval timeRange=case(daysToExpire>=120, "More than 120 Days", daysToExpire<120 AND daysToExpire>=90,

"90-120 Days", daysToExpire<90 AND daysToExpire>=60, "60-90 Days", daysToExpire<60 AND daysToExpire>=30, "30-60 Days", daysToExpire<30 AND daysToExpire>0, "Less than 30 Days", daysToExpire<=0, "Expired") | dedup Thumbprint | sort +daysToExpire | stats count by timeRange | rename count AS "Number of CAs" timeRange AS "Time Range"

DATA MODEL DRIVEN SPL

| tstats count FROM datamodel=Certificate_Authorities where earliest=-7d (X509_Certificate_Standard.StoreName="CA" OR X509_Certificate_Standard.StoreName="Root" OR X509_Certificate_Standard.StoreName="AuthRoot") by X509_Certificate_Standard.Thumbprint, X509_Certificate_Standard.Certificate_Authority_Stores.Days_Until_Expiration | eval timeRange=case('X509_Certificate_Standard.Certificate_Authority_Stores.Days_Until_Expiration'>=120, "More than 120 Days", 'X509_Certificate_Standard.Certificate_Authority_Stores.Days_Until_Expiration'<120 AND 'X509_Certificate_Standard.Certificate_Authority_Stores.Days_Until_Expiration'>=90, "90-120 Days", 'X509_Certificate_Standard.Certificate_Authority_Stores.Days_Until_Expiration'<90 AND 'X509_Certificate_Standard.Certificate_Authority_Stores.Days_Until_Expiration'>=60, "60-90 Days", 'X509_Certificate_Standard.Certificate_Authority_Stores.Days_Until_Expiration'<60 AND 'X509_Certificate_Standard.Certificate_Authority_Stores.Days_Until_Expiration'>=30, "30-60 Days", 'X509_Certificate_Standard.Certificate_Authority_Stores.Days_Until_Expiration'<30 AND 'X509_Certificate_Standard.Certificate_Authority_Stores.Days_Until_Expiration'>0, "Less than 30 Days", 'X509_Certificate_Standard.Certificate_Authority_Stores.Days_Until_Expiration'<=0, "Expired") | stats dc(X509_Certificate_Standard.Thumbprint) by timeRange | sort +timeRange | rename dc(X509_Certificate_Standard.Thumbprint) AS "Number of CAs" timeRange AS "Time Range"

Whitelist/Blacklist/Graylist

- Using the Blacklist.csv and Whitelist.csv lookups, determine whether the CA is on the blacklist, whitelist, both, or neither ("gray")
    a. White if on whitelist and not on blacklist: *isnotnull(white) AND isnull(black), "White"*
    b. Black if not on whitelist and on blacklist: *isnull(white) AND isnotnull(black), "Black"*
    c. Unknown ("gray") if on neither the whitelist nor the blacklist: *isnull(white) AND isnull(black), "Gray"*
    d. Both if on whitelist and on blacklist: *isnotnull(white) AND isnotnull(black), "Appears in Whitelist and Blacklist"*

REGULAR SPL

tag=certificate_index tag=certificate_source tag=certificate_authority hoursago=168 | eval ed2=strptime(ssl_end_time, "%m/%d/%Y %I:%M:%S %p") | eval now=now() | eval timeDiff=ed2-now | where timeDiff>0 | lookup "CA Whitelist" Good_Thumbprint AS Thumbprint OUTPUTNEW Good_Thumbprint AS white | lookup "CA Blacklist" Bad_Thumbprint AS Thumbprint OUTPUTNEW Bad_Thumbprint AS black | eval Status=case(isnotnull(white) AND isnull(black), "White", isnull(white) AND isnotnull(black), "Black", isnull(white) AND isnull(black), "Gray", isnotnull(white) AND isnotnull(black), "Appears in Whitelist and Blacklist") | dedup Thumbprint | table Thumbprint, ssl_issuer, ssl_subject, Status | rename ssl_issuer AS Issuer ssl_subject AS Subject

PIVOT SPL

| pivot Certificate_Authorities Certificate_Authority_Stores values(ssl_issuer) AS "Issuer" values(ssl_subject) AS "Subject" values(White_Black_List_Status) AS "Status" SPLITROW Thumbprint AS Thumbprint SORT 10000 Thumbprint ROWSUMMARY 0 COLSUMMARY 0 NUMCOLS 0 SHOWOTHER 1

Default vs. Non-Default

- Using the DefaultCAList.csv lookup file, determines if the CA has a browser identified and if so mark as "Default", else mark as "Non-default": *lookup DefaultCAList.csv "SHA-1 Fingerprint" AS Thumbprint | eval Status=if(isnull(Browser),"Non-default","Default")*

REGULAR SPL

tag=certificate_index tag=certificate_source tag=certificate_authority hoursago=168 | lookup DefaultCAList.csv "SHA-1 Fingerprint" AS Thumbprint | eval Status=if(isnull(Browser),"Non-default","Default") | chart dc(Thumbprint) AS "Number of CAs" by Status

DATA MODEL DRIVEN SPL

| tstats dc(X509_Certificate_Standard.Thumbprint) FROM datamodel=Certificate_Authorities where earliest=-7d (X509_Certificate_Standard.StoreName="CA" OR X509_Certificate_Standard.StoreName="Root" OR X509_Certificate_Standard.StoreName="AuthRoot") by X509_Certificate_Standard.Certificate_Authority_Stores.Default_Status | rename dc(X509_Certificate_Standard.Thumbprint) AS "Number of CAs" X509_Certificate_Standard.Certificate_Authority_Stores.Default_Status AS Status

Commonality of Number of Unique CAs per Host

1. Determine number of unique CAs per host: *stats dc(Thumbprint) AS dcount by host*
2. Determine number of hosts per unique CA: *stats count by dcount*

REGULAR SPL

tag=certificate_index tag=certificate_source tag=certificate_authority hoursago=168 | eval ed2=strptime(ssl_end_time, "%m/%d/%Y %I:%M:%S %p") | eval now=now() | eval timeDiff=ed2-now | where timeDiff>0 | stats dc(Thumbprint) AS dcount by host | stats count by dcount | sort dcount | rename count AS "Number of Hosts", dcount AS "Number of Unique CAs"

DATA MODEL DRIVEN SPL

| tstats dc(X509_Certificate_Standard.Thumbprint) AS dcount FROM datamodel=Certificate_Authorities where earliest=-7d (X509_Certificate_Standard.StoreName="CA" OR X509_Certificate_Standard.StoreName="Root" OR X509_Certificate_Standard.StoreName="AuthRoot") X509_Certificate_Standard.Certificate_Authority_Stores.Expired=False by host | stats count by dcount | sort dcount | rename count AS "Number of Hosts", dcount AS "Number of Unique CAs"

Hosts per CA

- Determine number of unique hosts per CA: *stats dc(host) AS "Number of Hosts" by Thumbprint*

REGULAR SPL

tag=certificate_index tag=certificate_source tag=certificate_authority hoursago=168 | eval ed2=strptime(ssl_end_time, "%m/%d/%Y %I:%M:%S %p") | eval now=now() | eval timeDiff=ed2-now | where timeDiff>0 | stats dc(host) AS "Number of Hosts" by Thumbprint

DATA MODEL DRIVEN SPL

| tstats dc(host) AS "Number of Hosts" FROM datamodel=Certificate_Authorities where earliest=-7d (X509_Certificate_Standard.StoreName="CA" OR X509_Certificate_Standard.StoreName="Root" OR X509_Certificate_Standard.StoreName="AuthRoot") X509_Certificate_Standard.Certificate_Authority_Stores.Expired=False by X509_Certificate_Standard.Thumbprint | rename X509_Certificate_Standard.Thumbprint AS Thumbprint

CA Algorithms

- Determine signature algorithm and key size for each unique CA: *chart dc(Thumbprint) by SignatureAlgorithm_FriendlyName, PublicKey_key_keysize*
- Note: CAs using elliptic curve cryptography algorithms (ECDSA or ECDH) do not include key sizes and so are marked as key size = 0.

REGULAR SPL

tag=certificate_index tag=certificate_source tag=certificate_authority hoursago=168 | eval ed2=strptime(ssl_end_time, "%m/%d/%Y %I:%M:%S %p") | eval now=now() | eval timeDiff=ed2-now | where timeDiff>0 | eval Public_Key_Size=coalesce(Public_Key_Size, "Unknown") | chart dc(Thumbprint) by ssl_signature_algorithm, Public_Key_Size

DATA MODEL DRIVEN SPL

| tstats dc(X509_Certificate_Standard.Thumbprint) AS dcount FROM datamodel=Certificate_Authorities where earliest=-7d (X509_Certificate_Standard.StoreName="CA" OR X509_Certificate_Standard.StoreName="Root" OR X509_Certificate_Standard.StoreName="AuthRoot") X509_Certificate_Standard.Certificate_Authority_Stores.Expired=False by X509_Certificate_Standard.ssl_signature_algorithm X509_Certificate_Standard.Certificate_Authority_Stores.Normalized_Key_Size | xyseries X509_Certificate_Standard.ssl_signature_algorithm X509_Certificate_Standard.Certificate_Authority_Stores.Normalized_Key_Size dcount | rename X509_Certificate_Standard.Certificate_Authority_Stores.Normalized_Key_Size AS "Key Size" dcount AS "Number of CAs" X509_Certificate_Standard.ssl_signature_algorithm AS "Signature Algorithm"