Beyond Cool Dashboards and Fancy Visualizations

Splunk as a full stack application development platform

Tieu Luu | VP Product Development, Qmulos

Daniel Letchev | Senior Software Engineer, Qmulos

October 2018

Forward-Looking Statements

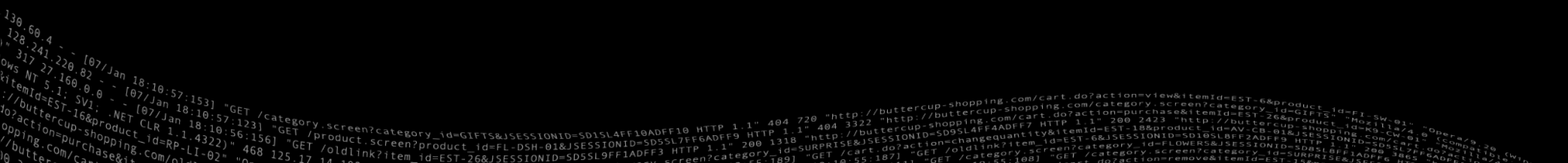
During the course of this presentation, we may make forward-looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC.

The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

splunk> .conf18

Agenda

- Qmulos Background
- Splunk Apps for Compliance and Auditing Automation
  - Q-Compliance
  - Q-Audit
- Features and Functionality of Q-Compliance
- Anatomy of an Enterprise Web Application
- Implementing each Subsystem/Component in Splunk
- Comparison with Building from Scratch/Open Source
- Key Takeaways
- Links to Useful Resources



Who Are We?

- Tieu Luu
- VP Product Development, Qmulos
  - 20+ years software development in government, enterprise IT, and startups
  - Last ten years focused on building cyber security analytics and continuous monitoring solutions for large enterprises
- tieu@qmulos.com
- @TechTieu

- Daniel Letchev
- Senior Software Developer, Qmulos
  - Develops Splunk apps @Qmulos that automate cyber security compliance and continuous monitoring
  - Prior to Qmulos, developed DoD's enterprise continuous monitoring solution that monitored the security posture of millions of devices across the DoD's networks
- daniel@qmulos.com

Qmulos Company Overview

Company

- Founded 2012 by DHS Exec responsible for .GOV Cybersecurity Automation
- Decades of proven IT compliance and cyber experience
- SDVOSB, TS facility clearance, 95% cleared staff, located in NoVa
- Reseller, PS, & ISV Splunk Partner

Products

- **Q-Audit**
  - ICS 500-27
  - Insider Threat
- **Q-Compliance**
  - NIST & Custom Controls
  - Various Frameworks
- Built as Premium Apps for the Splunk Platform
- Enables massive scale while automating security and compliance activities

Customers

- Includes small, medium, and large organizations
- Fortune 1000, Federal Civilian, DoD, & Intel Community customers
- **Focused on compliance automation to deliver operational security**

Government Innovation Awards

2018 Inc. 5000 LIST

Qmulos Splunk Apps

Cyber security compliance automation and auditing on the Splunk platform

splunk>

Q-Compliance

Visualizations

Analytics

Data Models

Q-Audit

Visualizations

Analytics

Data Models

HR systems
Training/ Learning management systems

Project planning/ management tools

Source code repositories
Code scanners

Agile lifecycle management tools
Automated testing tools

HVAC
Environmental sensors

Trouble ticketing systems

Incident management/ reporting systems

Operating systems
System logs

Asset management systems

Firewalls
Switches

CMDBs
Email filters/gateways

Data loss prevention systems

Storage systems
Backup solutions

Disaster recovery tools

Identity and access management solutions

Network IDS/NIPS
Host IDS/IPS

Perimeter defense solutions

Vulnerability/ threat intelligence tools

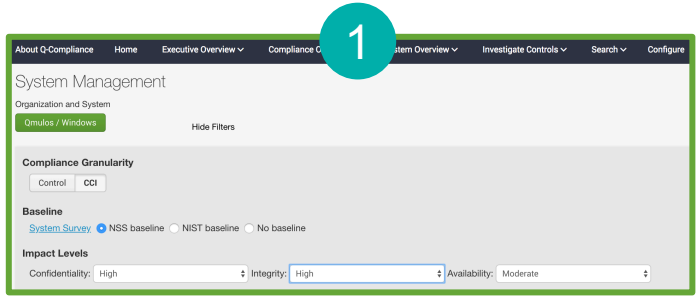
Endpoint protection tools

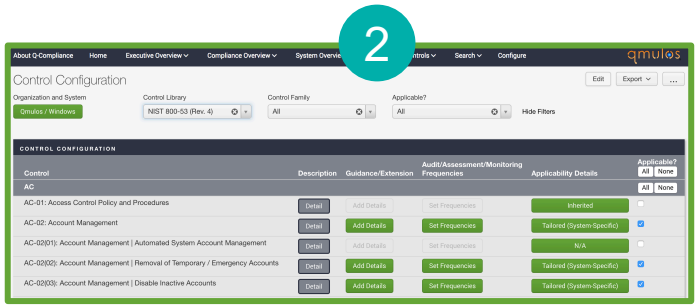
Vulnerability scanners

Patch management systems

splunk> .conf18

Q-Compliance – Workflow & Functionality





Categorize System

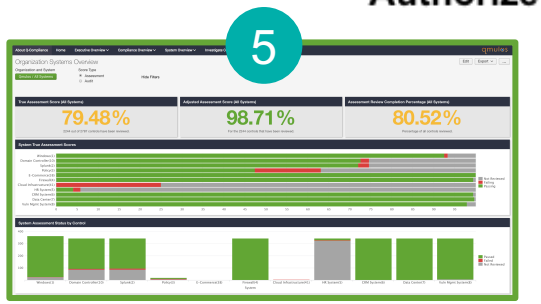
Monitor Controls

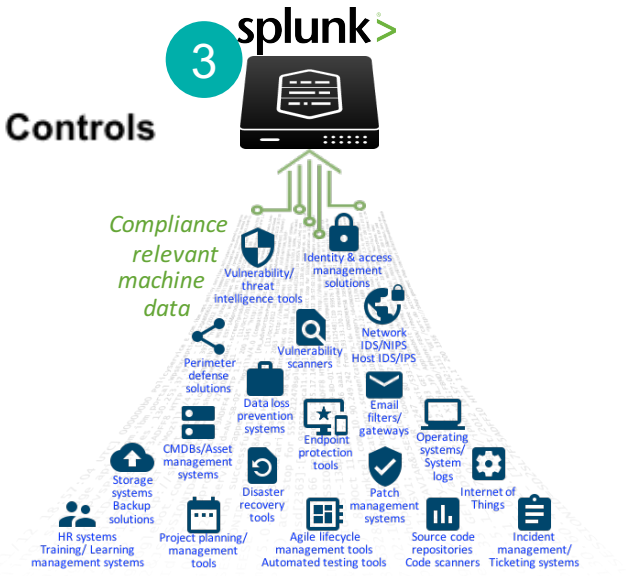
Select Controls

Authorize System

Implement Controls

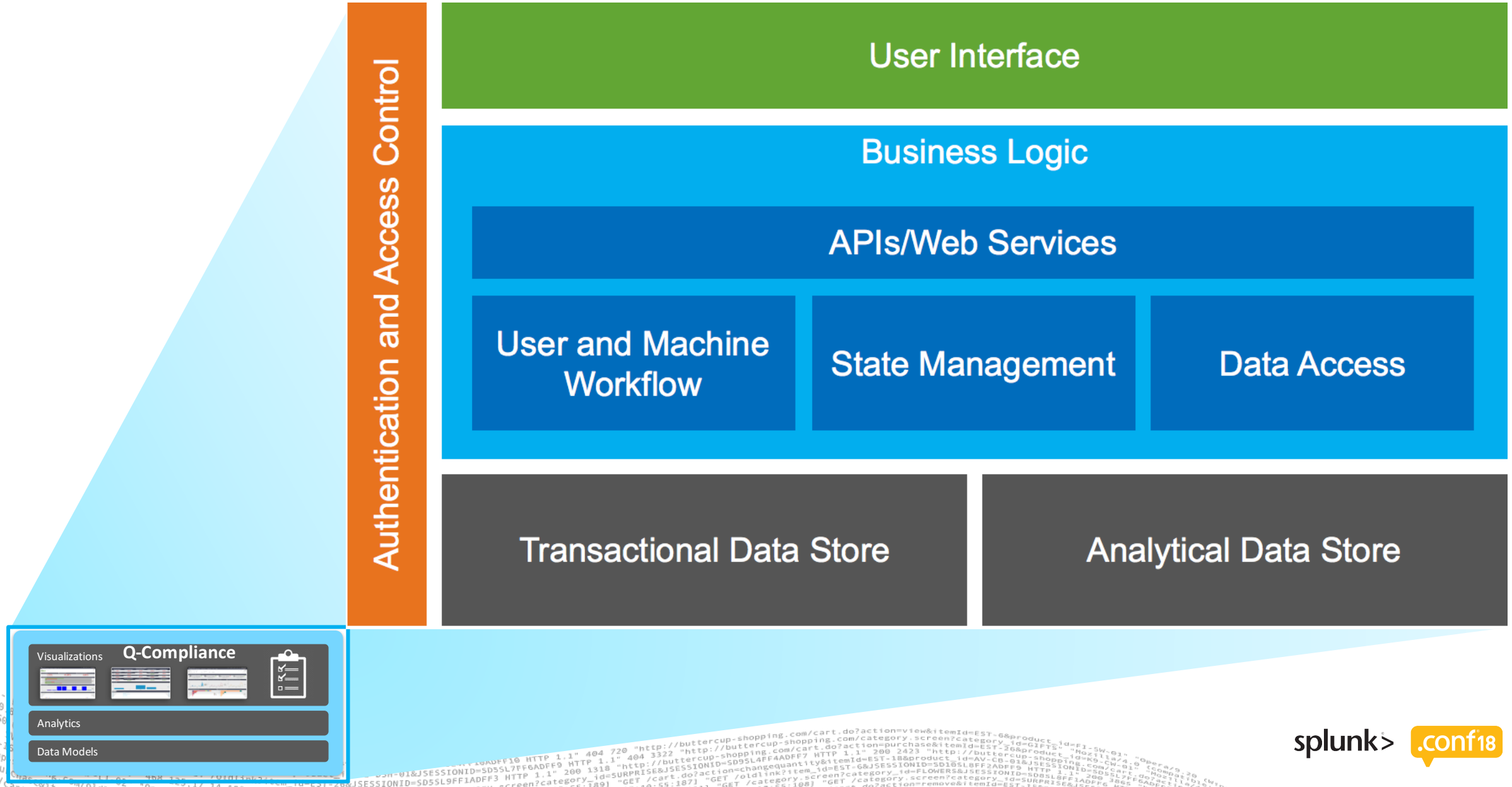
Assess Controls



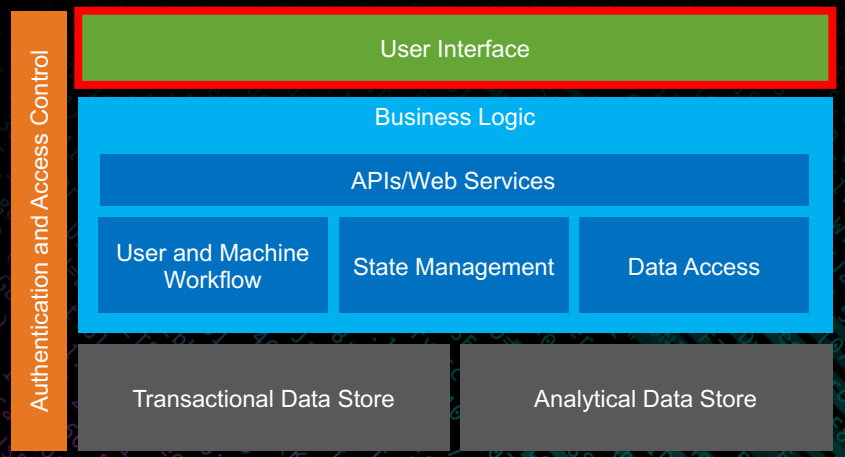


splunk> .conf18

Anatomy of an Enterprise Web Application



User Interface



1. Functionality

- Forms and modal dialogs to capture RMF user input, e.g. assessment findings, audit findings, system boundaries, organization-defined parameters, etc.
- Custom input elements to navigate hierarchical data structures, e.g. org structures
- Control dashboards with dynamic visualizations that vary based on the control selected

2. Implementation on Splunk

- Highly customized JavaScript and SimpleXML
  - Pages have a companion JavaScript file
  - HTML templates and <html> SimpleXML tags to allow for modal dialogs and custom sections inside XML pages
  - Splunk Web Framework to create tables, charts, and input elements dynamically
  - jQuery, jsGrid and other JavaScript libraries to further improve the user experience

User Interface

SimpleXML and JavaScript based page

Scoring Overview and Categories

Edit | Export | ...

Organization and System

Qmulos / Windows

Score Type

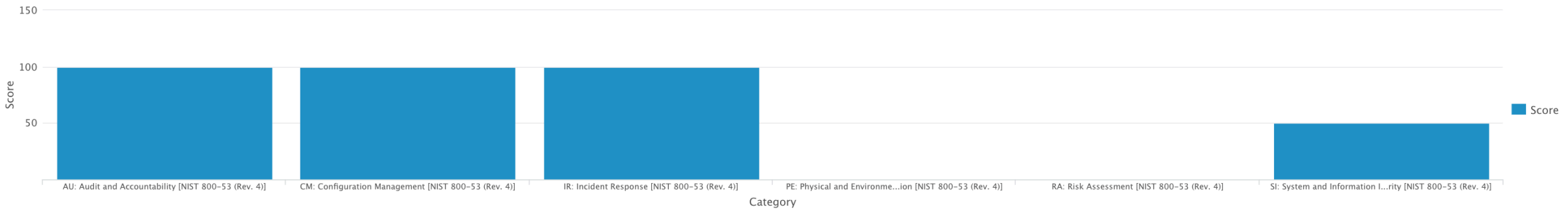
- (•) Assessment
- ( ) Audit

Hide Filters

FEDRAMP

DAILY	WEEKLY	10 DAYS	MONTHLY	60 DAYS
57.14%	100.00%	100.00%	72.73%	100.00%
90 DAYS	**ANNUALLY**	**2 YEARS**	**3 YEARS**	**5 YEARS**
75.00%	67.50%	100.00%	50.00%	0.00%

DAILY



DAILY

Control Name	Score
AU-02	100.00
CM-08(03)	100.00
IR-06	100.00

splunk> .conf18

User Interface

SimpleXML and JavaScript based page

SimpleXML

```
<fieldset autoRun="true" submitButton="false">
  <html>
    <div id="orgselect"></div>
  </html>
  <input id="typet" type="radio" token="type">
    …
</fieldset>
```

```
<row>
  <panel>
    <html>
      <div id="categories"></div>
    </html>
  </panel>
</row>
```

```
<row id="category" depends="$displaycat$">
  <panel>
    <table id="scoretable">
      <title>$category$</title>…
    </table>
  </panel>
</row>
```

JavaScript

Using JavaScript and HTML Template on page load

```
$("#orgselect").html(OrgSystemTemplate);
```

Using jQuery, SingleView and PostProcessManager

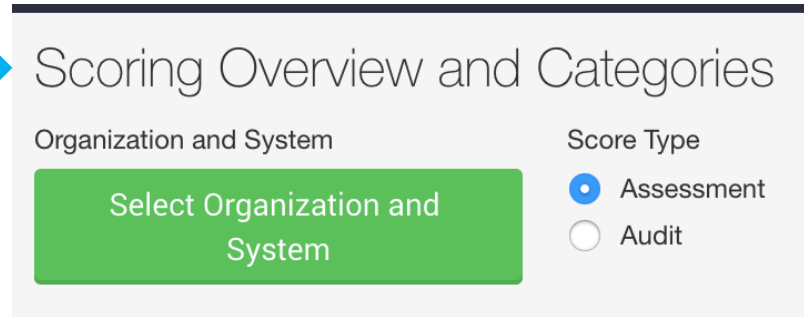
```
new PostProcessManager({…});
new SingleView({…});
```

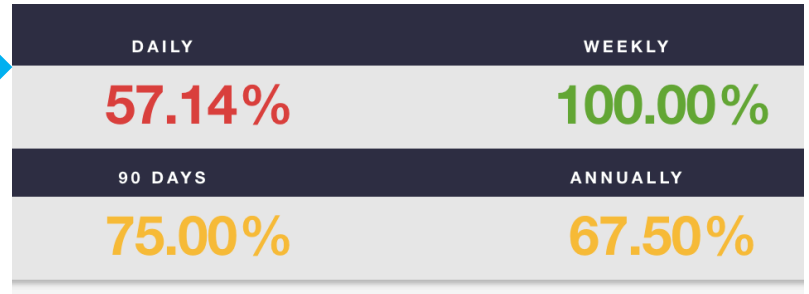
http://docs.splunk.com/Documentation/WebFramework

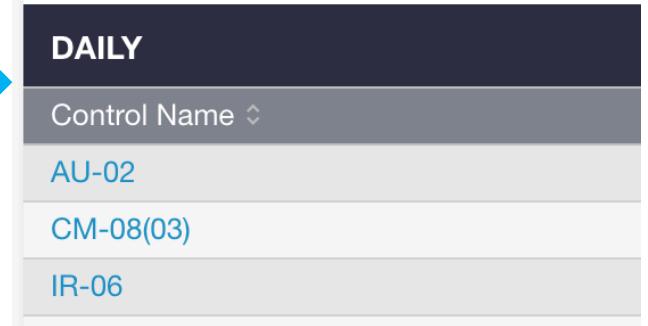
Setting tokens in the JavaScript

```
var tokens =
mvc.Components.get("default");
tokens.set("category", score_cat);
```

http://dev.splunk.com/view/webframework-developapps/SP-CAAAEW3

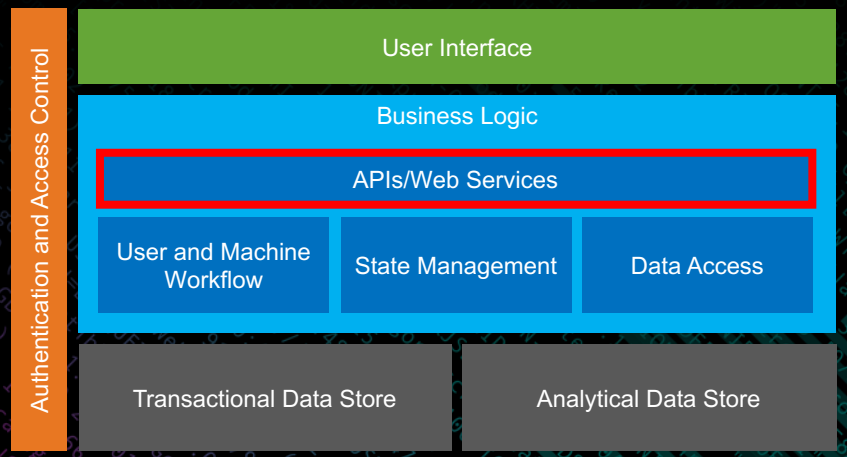






splunk> .conf18

APIs/Web Services



1. Functionality

- User interface access to business logic and data through APIs/web services, e.g.
  - Configure applicable controls and parameters for a system
  - Access system's compliance data
  - Upload/download compliance evidence
- Integration with 3rd party GRC tools
  - Share compliance data with other GRC tools in a customer's environment

2. Implementation on Splunk

- Implemented in Python with 2 approaches
  a. Custom cherrypy controllers
     - Part of the older Module System's MVC architecture
  b. Custom REST endpoints
     - Extending Splunk's REST API with custom endpoints
     - More "official", preferred way to do it

APIs/Web Services

Two options for implementing custom web services in Splunk

Custom cherrypy controller

web.conf

[endpoint:custom_controller]
- Expose the controller through Splunk Web at a particular URL path

custom_controller.py

- Implementation of the web service
- Definition of which python functions are exposed through which HTTP methods
- Inherits from controller.BaseController

- Good (but old) example of custom controllers on Splunk wiki: https://wiki.splunk.com/Community:40GUIDevelopment

Custom REST endpoint

restmap.conf

[script:custom_endpoint]
- Define the endpoint
- URL path to match
- Handler (python code) that implements the endpoint

web.conf

[expose:custom_endpoint]
- Expose the endpoint through Splunk Web at a particular URL path
- Supported HTTP methods

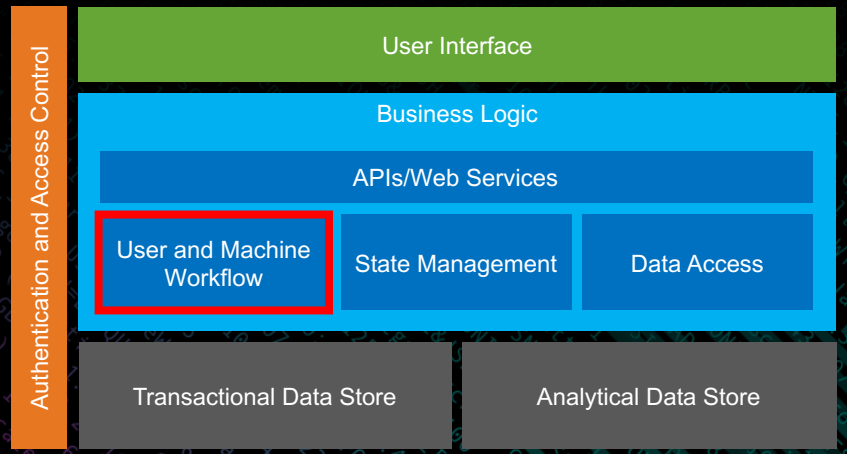
custom_endpoint.py

- Implementation of the web service
- inherits from splunk.rest.BaseRestHandler

- Excellent blog series from Hurricane Labs: https://www.hurricanelabs.com/splunk-tutorials/splunk-custom-endpoints-part-1-the-basics

splunk> .conf18

User and Machine Workflow



1. Functionality

- Interlinked screens to guide users through the 6 steps of the RMF process
- Dynamic custom drilldowns to guide users through different sub-flows within a given step
  - Reviewing a system-specific control vs. an inherited control from the Security Controls Traceability Matrix (SCTM) dashboard
- Custom alert actions to automate compliance assessments, notify users and invoke remediating actions

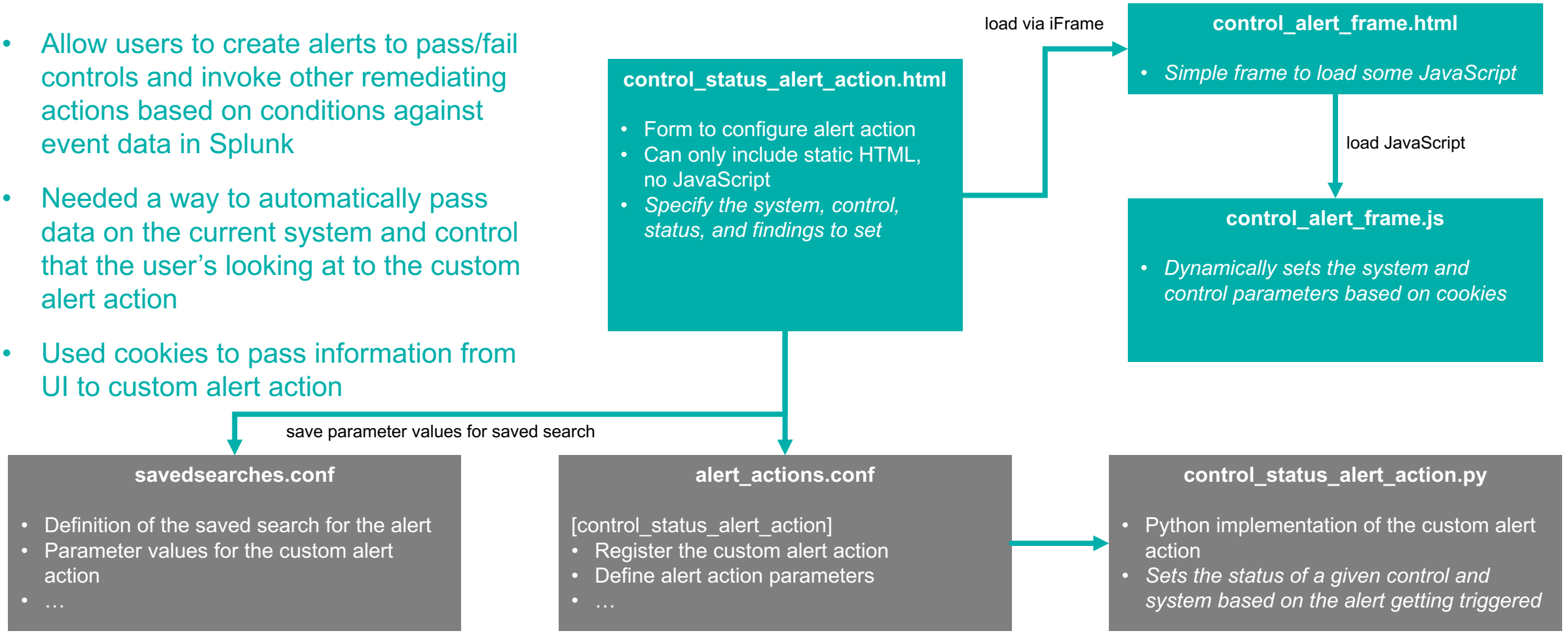
2. Implementation on Splunk

- Custom drilldowns to link dashboards together in a workflow (implemented in SimpleXML and JavaScript)
  - http://docs.splunk.com/Documentation/Splunk/latest/Viz/DrilldownIntro
- Tokens to pass state information across dashboards
- Custom alert action scripts (implemented in Python) that integrate with the standard Splunk alerting workflow
  - http://docs.splunk.com/Documentation/Splunk/latest/AdvancedDev/CustomAlertScript

User and Machine Workflow

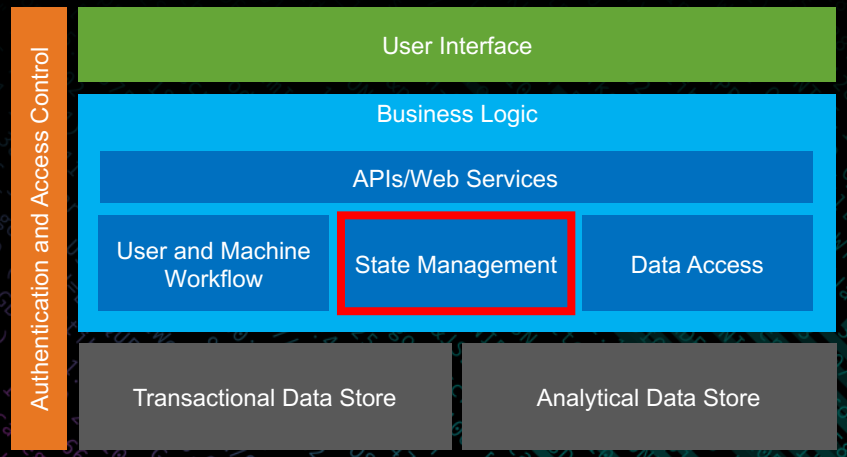
An example on how to integrate user and machine workflows

- Allow users to create alerts to pass/fail controls and invoke other remediating actions based on conditions against event data in Splunk
- Needed a way to automatically pass data on the current system and control that the user's looking at to the custom alert action
- Used cookies to pass information from UI to custom alert action



splunk> .conf18

State Management



1. Functionality

- Manage and store the state of key entities as they transition through the workflows:
  - Organizations
  - Systems
  - Controls
- Maintain the state of user interactions as they navigate across different dashboards and pages, e.g. currently selected orgs, systems, controls

2. Implementation on Splunk

- All dashboards, pages, web services are stateless, i.e. we don't maintain any session state
- Custom drilldowns and tokens are used to pass state data across different dashboards and pages
- All state data is stored in cookies and our transactional data store, i.e. KV store
  - Cookies to keep track of currently selected orgs and systems
  - System's state within the RMF process stored across multiple tables in transactional data store

State Management

Store state data in cookies and populate tokens with cookie values

Use js-cookies to manage cookies in JavaScript:
https://github.com/js-cookie/js-cookie

control_dashboard.js

```
require(["js-cookie", "splunkjs/mvc"],
  function(Cookie, mvc) {
    $(document).ready(function () {
        var setSystemCookie = function (value) {
            if (value && (value !== "*")) {
                Cookie.set("Q-Compliance_system", value);
            } else {
                Cookie.remove("Q-Compliance_system");
            }
        };

        var getSystemCookie = function () {
            var value = Cookie.get("Q-Compliance_system");
            if (!value) {
                value = undefined;
            }
            return value;
        };

        var tokens = mvc.Components.get("default");
        tokens.set("system", getSystemCookie());
    });
});
```

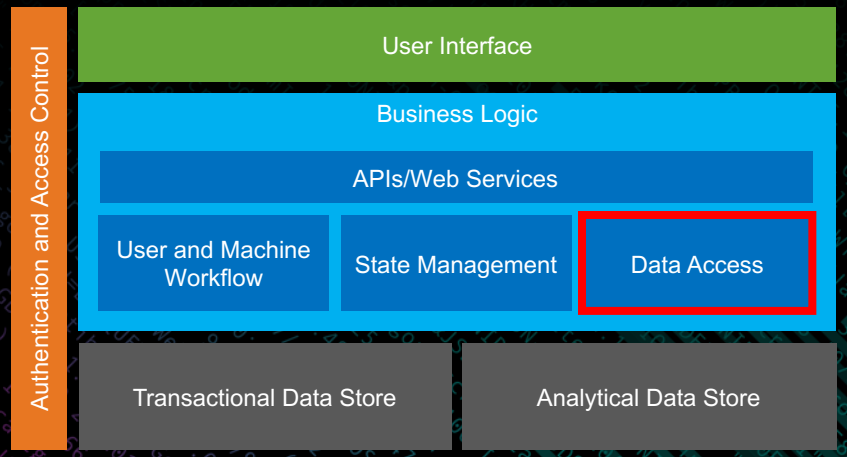
control_dashboard.xml

```
<dashboard script="control_dashboard.js">
    <row>
        <panel>
            <chart>
                <search>
                    <query>
                        | pivot Q_Account_Management
Account_Management dc(user) AS "Number of Accounts"
SPLITROW action AS action FILTER system is $system$ SORT
100 action ROWSUMMARY 0 COLSUMMARY 0 SHOWOTHER 1
                    </query>
                </search>
            </chart>
        </panel>
    </row>
</dashboard>
```

Filter the query in the dashboard with a system value that the user previously selected on another page

splunk> .conf18

Data Access



1. Functionality

- Access to various technical control evidence in the form of raw event data and accelerated data models
- Create/Read/Update/Delete access to RMF objects (e.g. organizations, systems, controls) through the KV stores

2. Implementation on Splunk

- Three different way of accessing data:
  - Through search queries directly embedded in the SimpleXML
  - Through JavaScript AJAX calls that use the Splunk REST endpoints for CRUD KV store operations or the Splunk search endpoint to retrieve more complex data
  - Through the Python SDK inside backend web services and custom alert actions

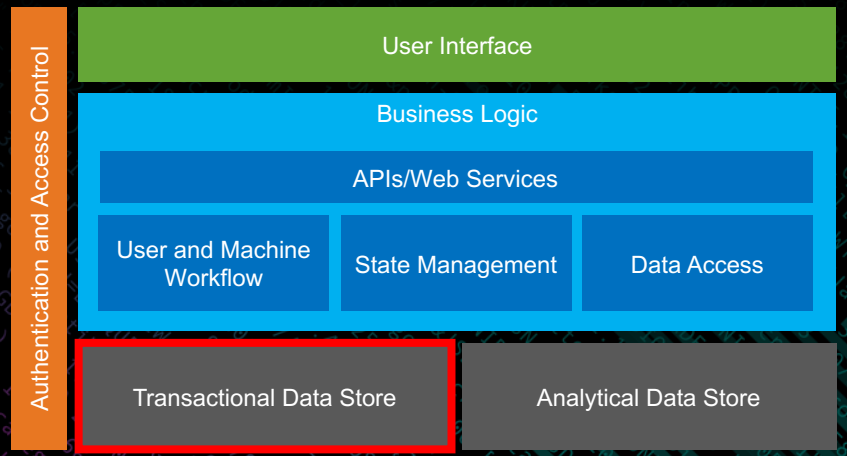
Data Access

The three different ways of accessing data

Data Access Type	Example	Pros	Cons
SimpleXML Query	<query> |inputlookup q-compliance_system_info</query>	• Easy to create • Easy to update in the XML	• Limited to mostly reads • Editing the KV store is difficult through search queries (outputlookup)
JavaScript using Splunk Data Endpoints	Splunk.util.make_url("/splunkd/__raw/servicesNS/nobody/Q-Compliance/storage/collections/data/kv_system_info); http://dev.splunk.com/view/webframework-developapps/SP-CAAAEZG	• Can do direct CRUD operations	• Requires a JavaScript file which moves the edits away from the XML
Python SDK	client.kvstore['kv_system_info’]; http://dev.splunk.com/python	• Data Layer separation • Allows for API with other apps	• Requires a custom endpoint

splunk> .conf18

Transactional Data Store



1. Functionality

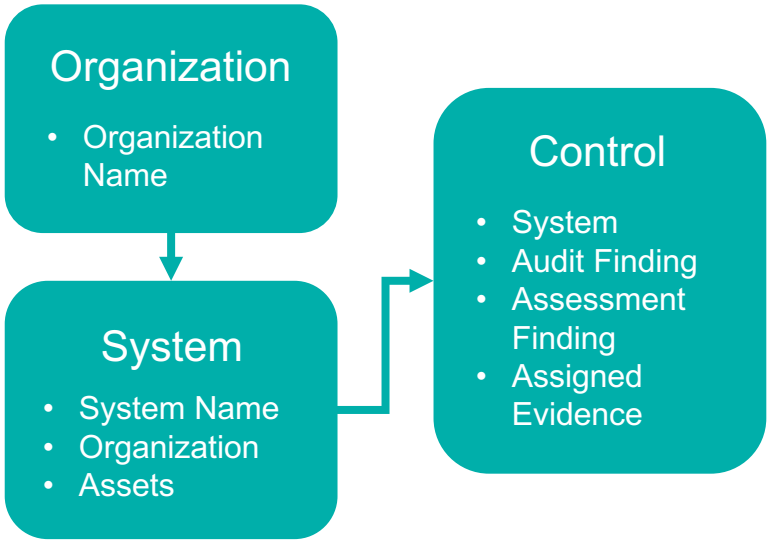
- Full blown database to store key RMF entities (e.g. organizations, systems, controls)
- Store the state of key entities and user interactions as they progress through different workflows
- Allow application customers to have a unique environment that represents their own security posture, important findings, and dependencies

2. Implementation on Splunk

- KV store implemented as a traditional entity-relationship structure, not just for lookups
- Create default datasets and have procedures for data migrations, application upgrades, and backwards compatibility

Transactional Data Store

Using KV store as a truly transactional data store

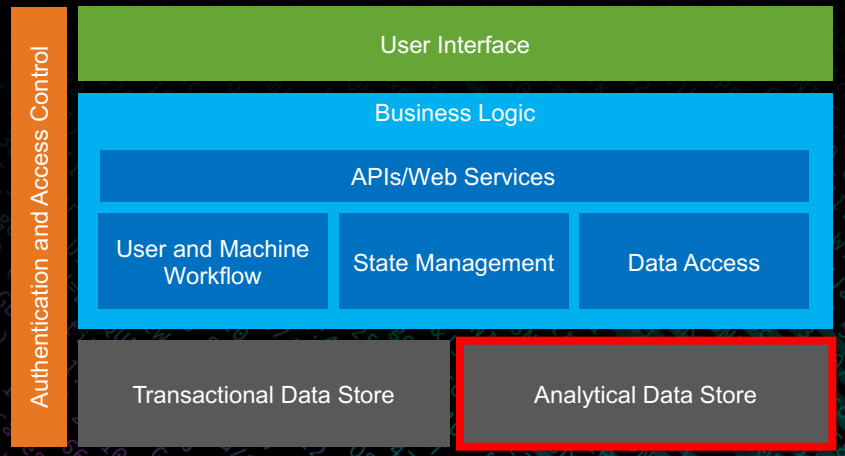


Maintenance Scenarios (in Splunk):

- Populating the Organization and System entities
  - Create the entities in collections.conf and transforms.conf
  - Create a User Interface which writes to the KV store using the Splunk endpoints or outputlookup search queries
  - Create a default dataset by making CSV lookups of organizations and systems and use saved searches to write to the KV store (|inputlookup ~.csv | outputlookup …)
- Updating the Control entity to include Next Assessment Date
  - Update the entity structure by editing the collections.conf and the transforms.conf
  - Update the User Interface to write the Next Assessment Date when editing a new Control
  - Create a data migration saved search that adds a default Next Assessment Date for all existing records (|inputlookup …| eval Next_Assessment_Date=..| outputlookup…)
- Add functionality to remove Controls for Systems that are deleted
  - Update the User Interface to remove Control entities when removing Systems
  - Create a purge script that removes Controls for System entities that don't exist (|inputlookup … | join … | where … | outputlookup …)

splunk> .conf18

Analytical Data Store



1. Functionality

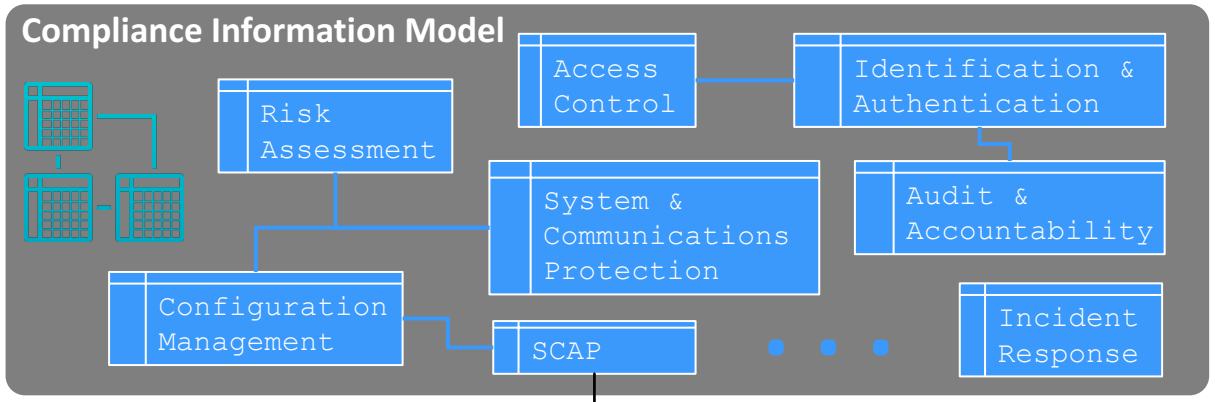
- Stores raw events, object-level details
- Drives the analytics and visualizations that assist with assessing control compliance
- Data models specific to security domains covered by NIST 800-53 controls – *"Qmulos Compliance Information Model (Q-CIM)"*

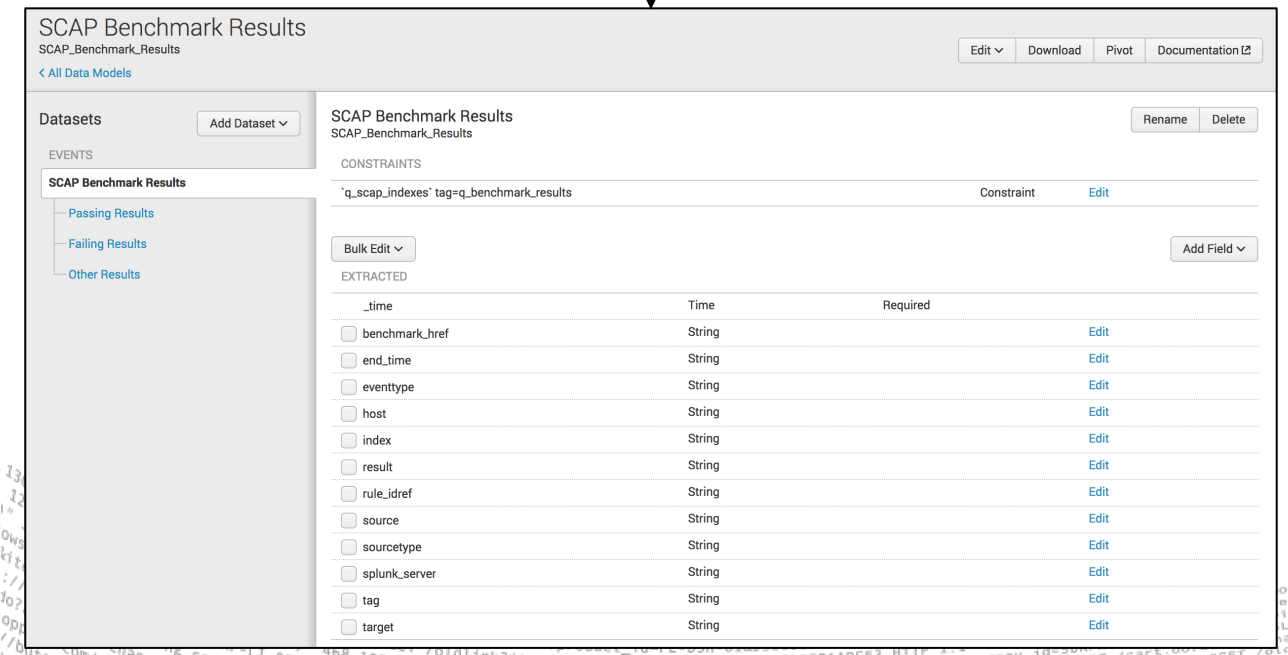
2. Implementation on Splunk

- Event data stored in indexes
- Accelerated data models
  - http://docs.splunk.com/Documentation/Splunk/latest/Knowledge/Aboutdatamodels
- Extension of Splunk's Common Information Model where relevant
  - http://docs.splunk.com/Documentation/CIM/4.11.0/User/Overview

Analytical Data Store

Things to consider when working with Splunk datamodels

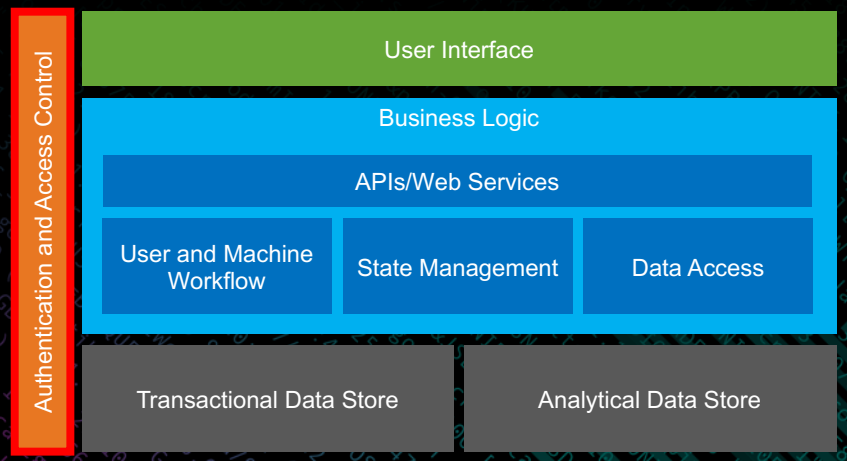




- Take a “waterfall approach” and do some rigorous analysis and design upfront
- Very disruptive to change your datamodels when customers and partners already have queries built using earlier versions
- Use Splunk CIM (or extend from it) as much as possible – makes data onboarding easier

splunk> .conf18

Authentication and Access Control



1. Functionality

- Capability access control: Restrict access to “controlled capabilities” (such as creating an assessment or deleting a system) by roles
- Data access control: Control who can see which system’s compliance and evidence based on what organizations they belong to

2. Implementation on Splunk

- Create roles that represent capabilities inside the application using authorize.conf (e.g. auditor, assessor)
- Add index role permissions to secure event data (e.g. an auditor might only see a summary index)
- Add collections permissions to default.meta to restrict read/write access to KV stores (e.g. only a system manager should be able to modify a system entity)
- Add views permissions to default.meta to restrict the pages that different roles can access (e.g. some pages should require a user to be admin)
- Use JavaScript to hide buttons/tables/other elements based on role to show a clean UI of what a user can do

Authentication and Access Control

Securing your Splunk web application

User Interface

- Access to Pages
  - Set page permissions for roles using default.meta
- Access to UI elements within a page
  - Get user roles through REST call
  - Hide the elements using custom JavaScript and CSS

Business Logic

- Restrict the data displayed in the User Interface to “subset” the user should see
  - Create custom REST endpoints to create the proper subsets of data
  - Add business logic to alert an user if they don’t have access and to narrow the input options (if they try to access data through a URL parameter)

Data Stores

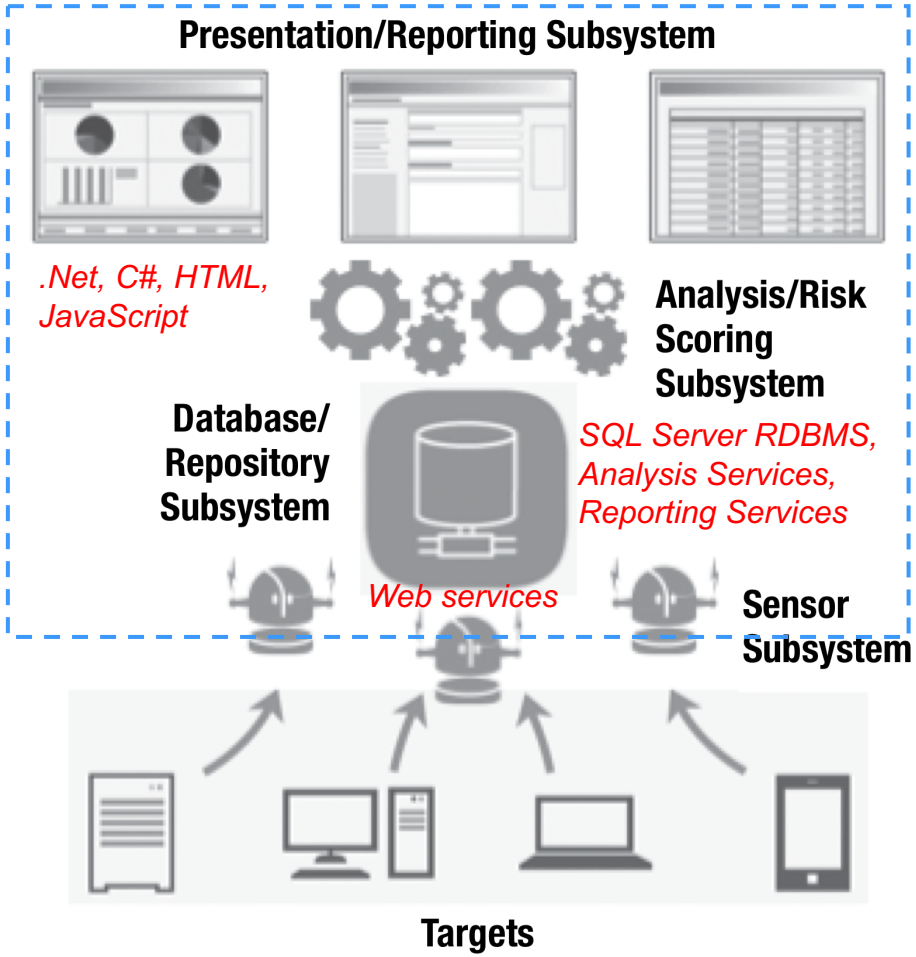
- Restrict access to KV Store
  - Set proper write permissions
  - Create a default role that has the basic read permissions
- Restrict access to Index data
  - Add index permissions to particular roles
  - Separate data into organization based indexes

Authentication and Access Control

splunk> .conf18

Comparison with Building from Scratch/Open Source

Previous experience building an enterprise continuous monitoring system covering two million assets



- Initial implementation - .Net stack with SQL Server backend
- Eventually migrated to Java and Hadoop to scale out
- Charts, graphs, tables were all custom implemented
- Lots of time spent on data integration code
  - Limited number of data sources that were integrated
- Scalability was a challenge

Took more time and resources but had less functionality

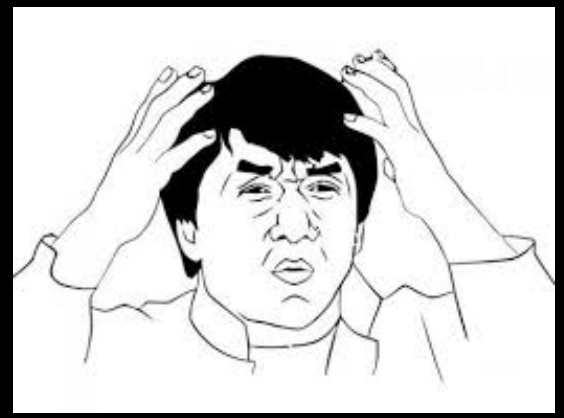
Case study in ISACA Journal: https://www.isaca.org/Journal/archives/2015/Volume-1/Pages/Implementing-an-Information-Security-Continuous-Monitoring-Solution.aspx

Key Takeaways

Splunk is a powerful platform for building modern enterprise web applications

- Tremendous productivity enhancer vs. building from scratch or using open source platforms but business logic and queries embedded in SimpleXML will only get you so far
- Do some upfront design and have a target architecture in mind, this will reduce refactoring pain later
- Apply common web architectures/design patterns, e.g. MVC (Model-View-Controller), CQRS (Command Query Responsibility Segregation), micro-services, etc.

Developing on top of Splunk

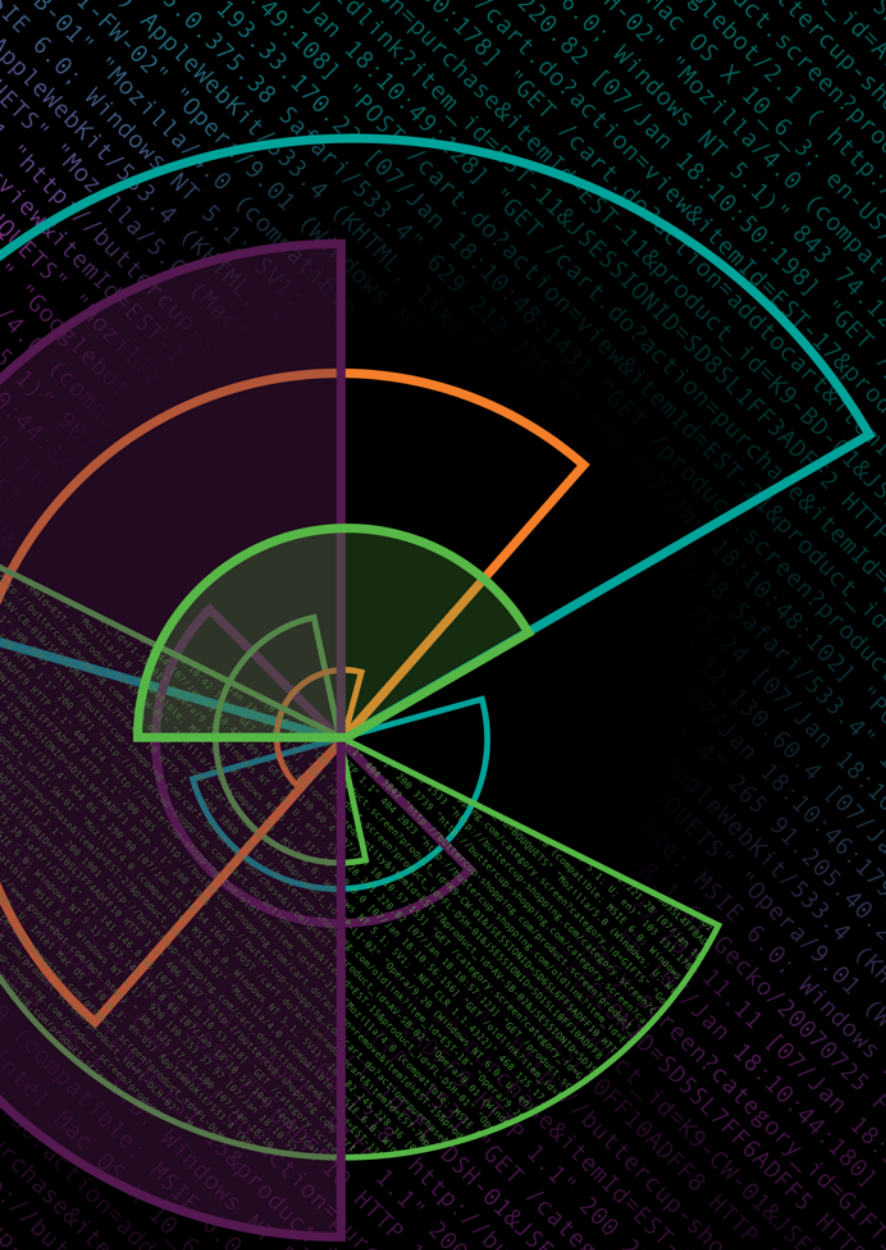


Developing from scratch/open source

Links to Useful Resources

Subsystem	Useful Resources for Implementation
User Interface	• Splunk Web Framework: http://docs.splunk.com/Documentation/WebFramework • Setting tokens in JavaScript: http://dev.splunk.com/view/webframework-developapps/SP-CAAAEW3
APIs/Web Services	• Custom controllers (the old way): https://wiki.splunk.com/Community:40GUIDevelopment • Custom REST endpoints (the new way): https://www.hurricanelabs.com/splunk-tutorials/splunk-custom-endpoints-part-1-the-basics
User and Machine Workflow	• Custom drilldowns: http://docs.splunk.com/Documentation/Splunk/latest/Viz/DrilldownIntro • Custom alert actions: http://docs.splunk.com/Documentation/Splunk/latest/AdvancedDev/CustomAlertScript
State Management	• KV Stores: http://dev.splunk.com/view/webframework-developapps/SP-CAAAEZK • JavaScript API for cookies: https://github.com/js-cookie/js-cookie
Data Access	• Accessing Splunk KV endpoints in JavaScript: http://dev.splunk.com/view/webframework-developapps/SP-CAAAEZG • Splunk Python SDK: http://dev.splunk.com/python
Transactional Data Store	• KV Stores: http://dev.splunk.com/view/webframework-developapps/SP-CAAAEZK
Analytical Data Store	• Splunk Datamodels: http://docs.splunk.com/Documentation/Splunk/latest/Knowledge/Aboutdatamodels • Common Information Model: http://docs.splunk.com/Documentation/CIM/4.11.0/User/Overview
Authentication and Access Control	• Creating roles in Splunk: https://docs.splunk.com/Documentation/Splunk/7.1.2/Security/Addandeditroleswithauthorizeconf • Set permissions for objects in Splunk: http://dev.splunk.com/view/webframework-developapps/SP-CAAAE88

splunk> .conf18



Q&A

www.qmulos.com

Thank You

Don't forget to rate this session in the .conf18 mobile app