

Integrating with Third-Party Tools using Splunk Alert Actions

Siegfried Puchbauer - Core Engineering, Splunk
Nicholas Filippi - Product Management, Splunk

.conf2016

splunk>

brought to you by...



Siegfried Puchbauer

- Lead Engineer, Dashboard Team
- Expert in Dashboards, Alerting, Javascript, and Bowler Hats



Nicholas Filippi

- Product Management, Splunk
- Responsible for Dashboards & Info Delivery

You are here because...

- Interested in connecting splunk to 3rd party systems
- Exploring the splunk developer platform
- Want to see something cool!
- Technical skills required
 - Development experience (preferably python)
 - Basic splunk app packaging experience

Agenda

- Alert Actions Framework Overview
- Demo
- Live Coding – Build alert action from scratch

Disclaimer

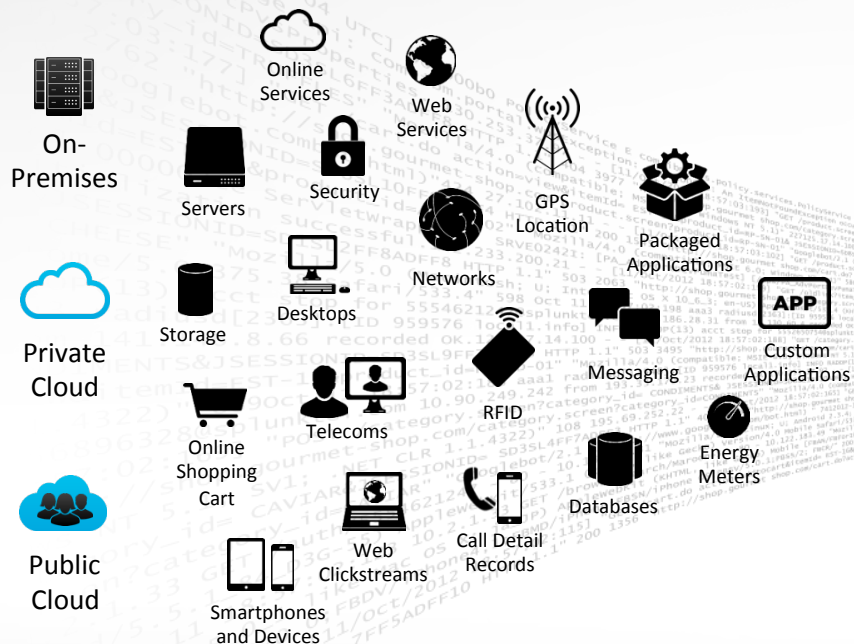
During the course of this presentation, we may make forward looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results 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 the 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

Make machine data accessible,
usable and valuable to everyone.

Turning Machine Data Into Business Value

Index Untapped Data: Any Source, Type, Volume



Ask Any Question

Application Delivery

IT Operations

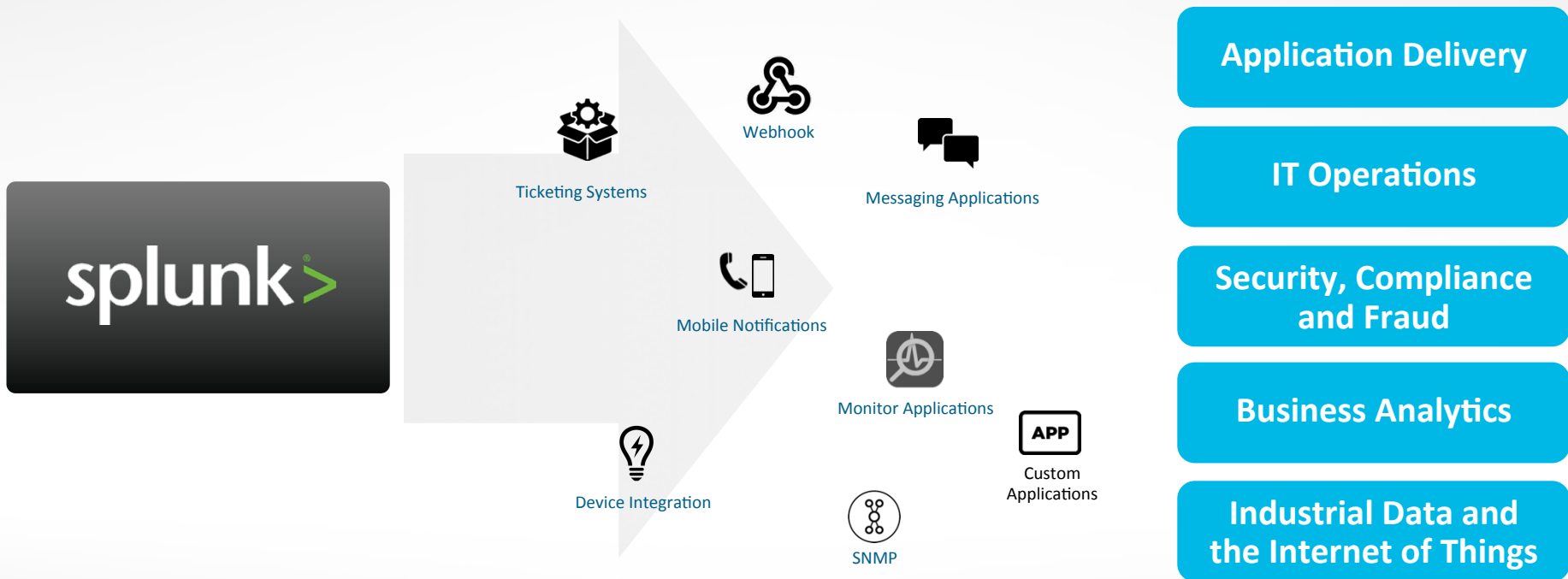
Security, Compliance
and Fraud

Business Analytics

Industrial Data and
the Internet of Things

Turning Machine Data Into Business Value

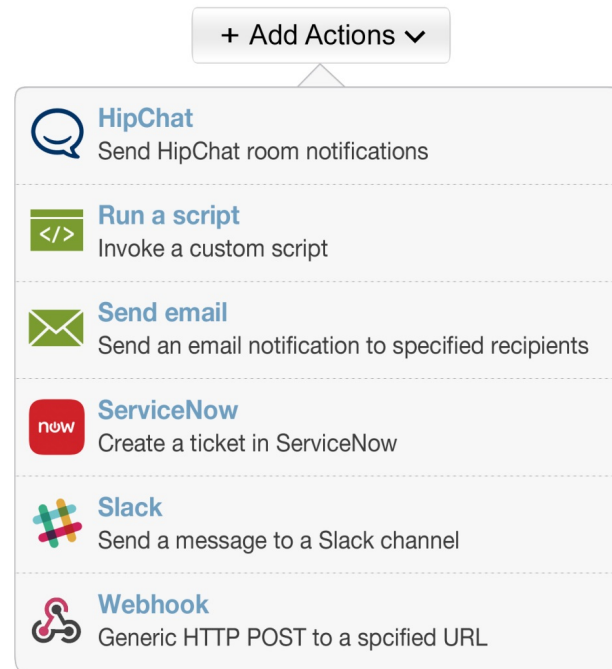
Integrate with other applications to automate workflows and improve efficiencies



Custom Alert Actions

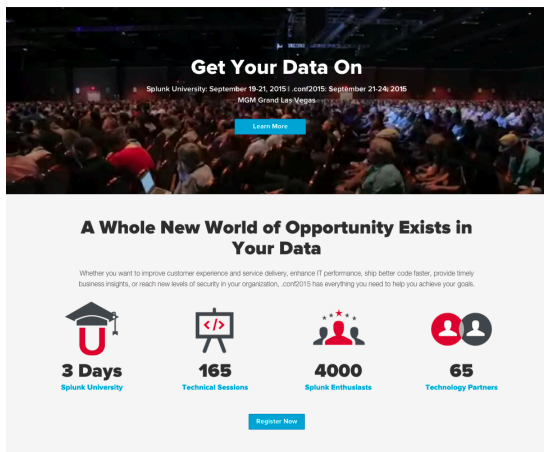
Use Splunk Alerts to trigger & automate workflows

- Allows packaged integration with third-party applications
- Simple admin/user configuration
- Developers can build, package, and publish alert action extensions for native integration to Splunk
- Growing list of integrations available



Real-world Example

Data Source



Monitoring mobile accelerometer data during keynote presentation.

Search & Trigger

Query:

```
index=main sourcetype=load  
| stats avg(load) as avg_load  
| search avg_load > 89
```

Use Splunk to trigger if average load exceeds threshold of 89

Alert Action



Trigger an air cannon to fire Splunk ponies into the crowd.

Alert Action Examples

- Notification Services
 - Send message to IM clients (HipChat, Slack)
 - Send SMS
- Incident Remediation / Ticketing
 - Automate the creation of tickets (ServiceNow, Jira)
- IT Monitoring
 - Send incident/alert into monitoring tools (xMatters, BigPanda)
- Security
 - Take action or send events to firewalls, devices, management consoles
- Internet-of-Things
 - Trigger device-level actions (change lights, sounds an alarm, send action to device)
- Custom Action
 - Trigger any organization-specific action (restart application, integrate with homegrown service, and more)

Eco-system Partners

(x) matters®

servicenow

octoblu

Now a part of Citrix

bigpanda

twilio

Alert Action Framework

✓	Packaging	Package and distribute custom alert actions within apps
✓	UI Integration	Build UI to support user input parameters within the alert workflow.
✓	Management	Admin-level management to enable/disable, view usage stats, and more
✓	Permissions	Access controls (configure role-level restrictions/permissions to access these alert actions)
✓	Logging	Logs both splunkd process level as well as script-level to internal index
✓	Input Validation	Perform user input validation on save of a given alert
✓	Dynamic Parameter Support (w/ token substitution)	Define parameters to be passed on script invocation; Available token substitution for 1 st result and search/job/server endpoints

Alert Action Framework

✓	Multi-platform Compatibility	Package multiple scripts to run based on platform type (Windows, Linux)
✓	Secured Storage of Credentials	Encrypt sensitive credentials on disk, with access methods to read/write from your alert script
✓	Ad-Hoc Invocation	Invoke directly from query string or workflow actions for testing or targeted use cases
✓	Setup/Configuration	Leverage the app setup.xml to persist global-level script parameters. These parameters can also be layered with invocation-specific parameters.
✓	Runtime Arguments (Language Agnostic)	Specify runtime parameters (ex., java interpreter and argument flags)

Demo!



Alert Action: From Scratch

.conf2016

Resources

Available information relevant for developers and admins

- Splunk Docs
 - <http://docs.splunk.com/Documentation/Splunk/6.3.0/AdvancedDev/ModAlertsIntro>
 - <http://docs.splunk.com/Documentation/Splunk/6.3.0/Alert/Setupalertactions#Webhooks>
- Splunkbase
 - Partner Apps
 - xMatters, BigPanda, Twilio, ServiceNow, Octoblu
 - Internal
 - Hipchat, Slack, Hue Bulbs
- Developer Guidance
 - Updated chapter on alert actions
 - Reference implementation – Atlassian Jira Integration



Wrap-Up

- Use Splunk Alerts to Automate Workflows & Integrate with External Applications
 - Notification/Messaging Services, Incident Remediation (Ticketing) Solutions, IT Monitoring Tools, Security Solutions, Internet-of-Things Devices, and Custom Applications
- Use Custom Alert Actions via the “Add Actions” scheduled alert menu
- Manage Alert Actions via “Settings > Alert actions” page
 - Find more alert actions to install via “Browse more”
- Leverage Docs/Examples/Developer-Guidance to Develop a Custom Alert Action

THANK YOU

.conf2016

Developer Guide

.conf2016

splunk>

Build a new alert action

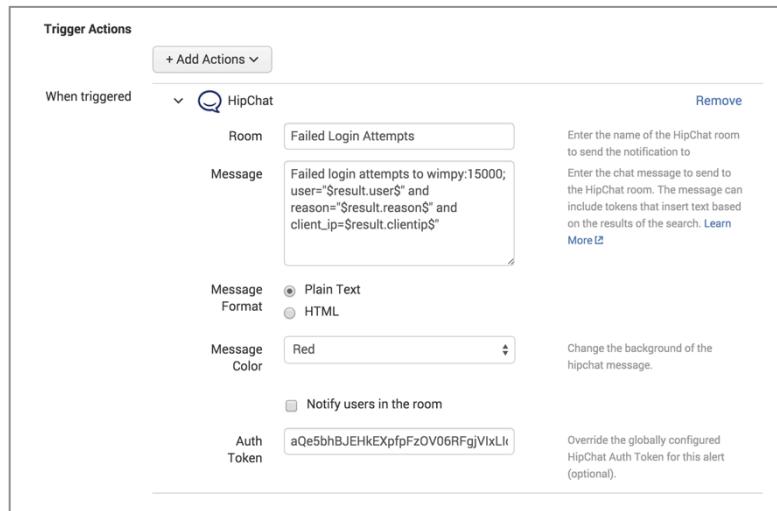
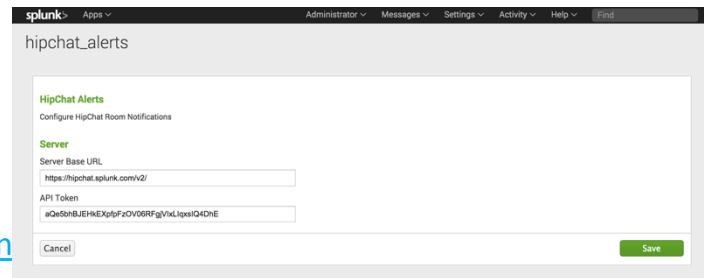
Steps

1. Alert action definition (alert_actions.conf)
2. Alert action script (in <app>/bin)
3. UI for configuring the alert action (HTML)
4. Spec for custom parameters
5. Optionally:
 - Icon
 - Validation rules
 - App setup



Example Alert Action

- Atlassian Hipchat (Group Chat)
- API v2 – Send Room Notifications
 - https://www.hipchat.com/docs/apiv2/method/send_room_notification
- Custom Alert Action – Includes:
 - App setup page to specify Hipchat server URL & API Token
 - For each alert, user should be able to configure
 - Room
 - Message
 - Format (plaintext, html)
 - Message color
 - Notify users?



Registration

default/alert_actions.conf

```
[hipchat]

is_custom = 1

label = HipChat
description = Send HipChat room notifications
icon_path = hipchat_alert_icon.png

payload_format = json

disabled = 0

param.base_url = https://api.hipchat.com/v2/
param.auth_token =
```



Trigger Actions

+ Add Actions ▾

- HipChat**
Send HipChat room notifications
- List in Triggered Alerts**
Make available in triggered alerts
- Run a script**
Invoke a custom script
- Send Email**
Send an email notification to specified recipients
- ServiceNow**
Create a ticket in ServiceNow
- Webhook**
Generic HTTP POST to a specified URL

Alert Script or Binary

- `<app>/bin/<name>`
 - **bin/hipchat.py**
- Carries out the actual alert action logic
- Receives XML or JSON payload from Splunk when invoked
- Supposed to be short-lived
 - Terminate as soon as action has been executed
- Python, shell scripts or binaries
- Multi-platform/architecture support

Execute a custom command

- Execution command/
arguments
 - Override default filename/path
 - Specify custom interpreter
 - Pass args

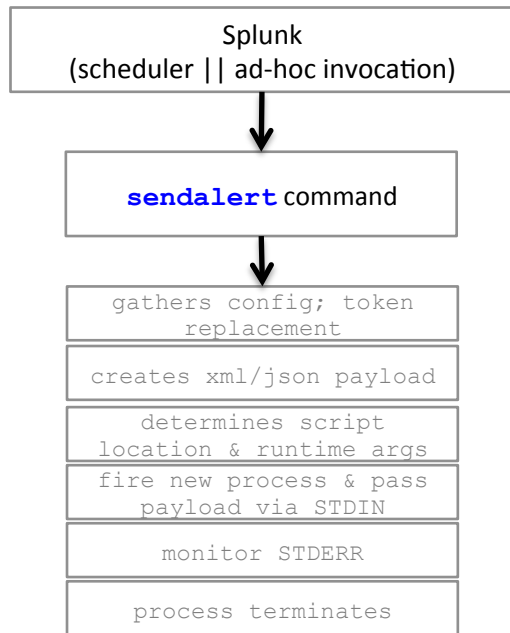
default/alert_actions.conf

```
[myaction]
...
alert.execute.cmd = java.path
alert.execute.cmd.arg.0 = -jar
alert.execute.cmd.arg.1 = $SPLUNK_HOME/etc/apps/myapp/bin/my.jar
alert.execute.cmd.arg.2 = --execute
```

Multi-Platform Compatibility

- Determines the alert script or binary to execute using the below precedence
 - \$SPLUNK_HOME/etc/apps/<app_name>/<arch>/bin
 - (arch : linux_x86_64, darwin_x86_64, windows_x86_64, windows_x86)
 - \$SPLUNK_HOME/etc/apps/<app_name>/bin

Script Invocation (Example)



```
{
  "server_host": "localhost:8089",
  "server_uri": "https://localhost:8089",
  "session_key": "1234512345",
  "results_file": "/opt/splunk/var/run/splunk/12938718293123.121/results.csv.gz",
  "results_link": "http://splunk.server.local:8000/en-US/app/search?sid=12341234.123",
  "sid": "12341234.123",
  "search_name": "My Saved Search",
  "owner": "admin",
  "app": "search",

  "configuration": {
    "room": "DevOps",
    "message": "Failed login attempts to wimpy:15000; user='Jeff' and reason='user-initiated' and client_ip='10.14.0.186'",
    "message_format": "plain",
    "base_url": "https://api.hipchat.com/v2/",
    "auth_token": "KJHJKHJGJHLKNJBLJBBL"
  },
  "result": {
    "sourcetype": "splunkd_access",
    "count": "24",
    "user": "Jeff",
    "client_ip": "10.14.0.186"
  }
}
```


Logging

- Messages printed to STDERR are logged to splunkd.log
- Direct access to alert action logs linked from “Alert actions” manager page (via “[view log events](#)”)
 - Includes both (a) splunkd logs for alert action, and (b) alert-specific log
- Search Query
 - `index=_internal sourcetype=splunkd component=sendmodalert action="<action_name>"`

The screenshot displays the Splunk search results for the query `index=_internal sourcetype=splunkd component=sendmodalert action="hipchat"`. The search returned 91 events. The interface includes a search bar, a timeline visualization, and a list of search results.

Search Query: `index=_internal sourcetype=splunkd component=sendmodalert action="hipchat"`

Results Summary: 91 events (before 5/11/15 8:13:11.000 PM)

Selected Fields: `host`, `source`, `sourcetype`

Interesting Fields: `action`, `color`, `component`, `date_hour`, `date_mday`

#	Time	Event
>	5/8/15 10:31:32.635 AM	05-08-2015 10:31:32.635 -0700 INFO sendmodalert - action=hipchat - Alert action script completed in duration=221 ms with exit code=0 host = wimpy.splunk.com source = /home/nfilippi/modalerts/splunk/var/log/splunk/splunkd.log sourcetype = splunkd
>	5/8/15 10:31:32.627 AM	05-08-2015 10:31:32.627 -0700 INFO sendmodalert - action=hipchat STDERR - Room notification successfully sent host = wimpy.splunk.com source = /home/nfilippi/modalerts/splunk/var/log/splunk/splunkd.log sourcetype = splunkd
>	5/8/15 10:31:32.627 AM	05-08-2015 10:31:32.627 -0700 INFO sendmodalert - action=hipchat STDERR - HipChat server responded with HTTP status=204 host = wimpy.splunk.com source = /home/nfilippi/modalerts/splunk/var/log/splunk/splunkd.log sourcetype = splunkd
>	5/8/15 10:31:32.466 AM	05-08-2015 10:31:32.466 -0700 INFO sendmodalert - action=hipchat STDERR - Sending message to hipchat room=Mod Alerts Ember Test with format=text host = wimpy.splunk.com source = /home/nfilippi/modalerts/splunk/var/log/splunk/splunkd.log sourcetype = splunkd
>	5/8/15 10:31:32.412 AM	05-08-2015 10:31:32.412 -0700 INFO sendmodalert - Invoking modular alert action=hipchat for search="Failed Login Attempts" sid="rt_scheduler__admin__search_RMD552557e8deaf61027_at_1430946360_5248.1" in app="search" owner="admin" type="saved" host = wimpy.splunk.com source = /home/nfilippi/modalerts/splunk/var/log/splunk/splunkd.log sourcetype = splunkd

UI Integration

default/data/ui/alerts/hipchat.html

```
<!-- Defines UI to be rendered in the alert workflow (w/ data binding to savedsearch.conf params) -->
```

```
<form class="form-horizontal form-complex">
  <div class="control-group">
    <label class="control-label" for="hipchat_room">Room</label>

    <div class="controls">
      <input type="text" name="action.hipchat.param.room" id="hipchat_room" />
      <span class="help-block">Enter the name of the HipChat room to send the notific
    </div>
  </div>
  <div class="control-group">
    <label class="control-label" for="hipchat_message">Message</label>

    <div class="controls">
      <textarea name="action.hipchat.param.message" id="hipchat_message"></textarea>
      <span class="help-block">
        Enter the chat message to send to the HipChat room.
        The message can include tokens that insert text based on the results of the search.
        <a href="{{SPLUNKWEB_URL_PREFIX}}/help?location=learnmore.alert.action.tokens"
target="_blank"
        title="Splunk help">Learn More <i class="icon-external"></i></a>
      </span>
    </div>
  </div>
  ...
</div>
```

The screenshot shows a configuration window for a HipChat alert. At the top, there's a '+ Add Actions' button and a 'Remove' link. Below, the 'HipChat' section is expanded, showing fields for 'Room' (a text input), 'Message' (a large text area), 'Message Format' (radio buttons for 'Plain Text' and 'HTML'), 'Message Color' (a dropdown menu currently set to 'None'), and 'Auth Token' (a text input containing a long alphanumeric string). There are also checkboxes for 'Notify users in the room' and 'Override the globally configured HipChat Auth Token for this alert (optional)'. On the right side, there are instructions: 'Enter the name of the HipChat room to send the notification to', 'Enter the chat message to send to the HipChat room. The message can include tokens that insert text based on the results of the search. Learn More', and 'Change the background of the hipchat message.'

Note: Currently, we only support static html (filter out any javascript). Markup should follow the bootstrap syntax (<http://getbootstrap.com/2.3.2/base-css.html#forms>)

Spec Files

README/alert_actions.conf.spec

```
# Defines parameters that to be added to  
the alert_actions.conf specification
```

```
[hipchat]
```

```
param.base_url = <string>
```

```
* HipChat API base URL - adjust if you're  
using you own server on premise
```

```
param.auth_token = <string>
```

```
* HipChat OAuth2 token  
* see https://www.hipchat.com/docs/apiv2/auth
```

README/savedsearches.conf.spec

```
# Defines parameters that to be added to the  
savedsearches.conf specification  
# HipChat alert settings
```

```
action.hipchat = [0|1]
```

```
* Enable hipchat notification
```

```
action.hipchat.param.room = <string>
```

```
* Name of the room to send the notification to
```

```
action.hipchat.param.message = <string>
```

```
* The message to send to the hipchat room.
```

```
...
```

Input Validation

Available via restmap.conf

```
[validation:savedsearch]
action.webhook.param.url = validate( match('action.webhook.param.url',
    "^https?://[^\s]+$"), "Webhook URL is invalid")
```

Save As Alert

 In handler 'savedsearch': Webhook URL is invalid

Settings

Title

Failed Logins - Webhook

Description

Optional

Permissions

Private

Shared in App

Setup & Configuration

- Available via app-level setup.xml
 - <http://docs.splunk.com/Documentation/Splunk/6.3.0/AdvancedDev/SetupApp>
 - <http://docs.splunk.com/Documentation/Splunk/6.3.0/AdvancedDev/SetupXML>
- Linked from “Alert actions” manager page

splunk> Apps ▾ Administrator ▾ Messages ▾ Settings ▾ Activity ▾ Help ▾ Find

hipchat_alerts

HipChat Alerts
Configure HipChat Room Notifications

Server

Server Base URL

API Token

Secure Storage of Credentials

- Leverages Splunk's storage/passwords endpoint for CRUD operations
 - <http://docs.splunk.com/Documentation/Splunk/6.2.3/RESTREF/RESTaccess#storage.2Fpasswords>
- Note: This can only be used for global-level alert action settings (not per-alert)
- Required Steps:
 - Add entry to app.conf
 - (ex. "[credential::hipchat_api_token:]")
 - Update setup.xml to write to storage/passwords endpoint
 - (ex. '<input endpoint="storage/passwords" entity=":hipchat_api_token:" field="password">')
 - Update script to get secure credentials from splunkd endpoint

default/bin/hipchat.py

```
...
def get_api_key(payload):
    url_tpl = '%s/servicesNS/nobody/hipchat_alerts/storage/passwords/%3Ahipchat_api_token%3A?output_mode=json'
    req = urllib2.Request(url_tpl % payload.get('server_uri'), None,
                          {'Authorization': 'Splunk %s' % payload.get('session_key')})
    res = urllib2.urlopen(req)
    body = json.loads(res.read())
    return body['entry'][0]['content']['clear_password']
...
```


Access Control

- Packaged alerts actions should set to global
 - Via `metadata/default.meta`

```
[alert_actions/hipchat]  
export = system
```

- Enable role-level permissions via “Alert actions” manager page

The screenshot shows the Splunk Alert Actions manager interface. On the left, a list of alert actions is displayed, including Adlassian Jira, FS Network Firewall Rule, HipChat, IBM Tivoli, Run a script, Send Email, ServiceNow, and Webhook. The 'HipChat' action is selected. A modal window titled 'Edit Permissions' is open, showing the configuration for the 'hipchat' alert action. The 'Owner' is set to 'nobody' and the 'App' is 'hipchat_alerts'. The 'Display For' dropdown is set to 'Owner'. The permissions table shows that 'Everyone' has 'Read' access (checked) and 'Write' access (unchecked). The 'admin' role has 'Read' access (unchecked) and 'Write' access (checked). The 'can_delete' role has 'Read' access (unchecked) and 'Write' access (unchecked). The 'power' role has 'Read' access (checked) and 'Write' access (unchecked). The 'splunk-system-role' has 'Read' access (checked) and 'Write' access (unchecked). The 'user' role has 'Read' access (checked) and 'Write' access (unchecked).

Role	Read	Write
Everyone	☒	☐
admin	☐	☒
can_delete	☐	☐
power	☒	☐
splunk-system-role	☒	☐
user	☒	☐

Packaging

- App can package more than 1 alert action
- Packaged files support
 - Registration as an alert action
 - Script itself
 - UI integration
 - Configuring alert/search parameters
 - Permissions (exporting to global)
 - Default param values
 - Etc.

Example folder/file structure of an app shipping a mod alert action

```
hipchat_alerts
├── appserver
│   └── static
│       ├── appIcon.png
│       └── hipchat_icon.png
├── bin
│   └── hipchat.py
├── default
│   ├── alert_actions.conf
│   ├── app.conf
│   ├── restmap.conf
│   ├── setup.xml
│   ├── data
│   │   └── ui
│   │       └── alerts
│   │           └── hipchat.html
│   ├── README
│   ├── alert_actions.conf.spec
│   └── savedsearches.conf.spec
└── metadata
    └── default.meta
```

THANK YOU

.conf2016