Copyright © 2015 Splunk Inc.

.conf2015

# Using Web Logs in Splunk to Dynamically Create Synthetic Transaction Tests

Justin Brown

IT Engineer

Pacific Northwest National Laboratory

splunk>

# 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.

# Agenda

- Introduction to Selenium and existing Splunk app
- Demo of data collected
- Building manual tests in Python
- Problems with the manual method
- Building dynamic tests using Splunk data
- Alerting on issues
- Ideas for expanding


# Selenium Testing

splunk>

# Introduction to Selenium

- Selenium – Web Browser Automation
  http://www.seleniumhq.org

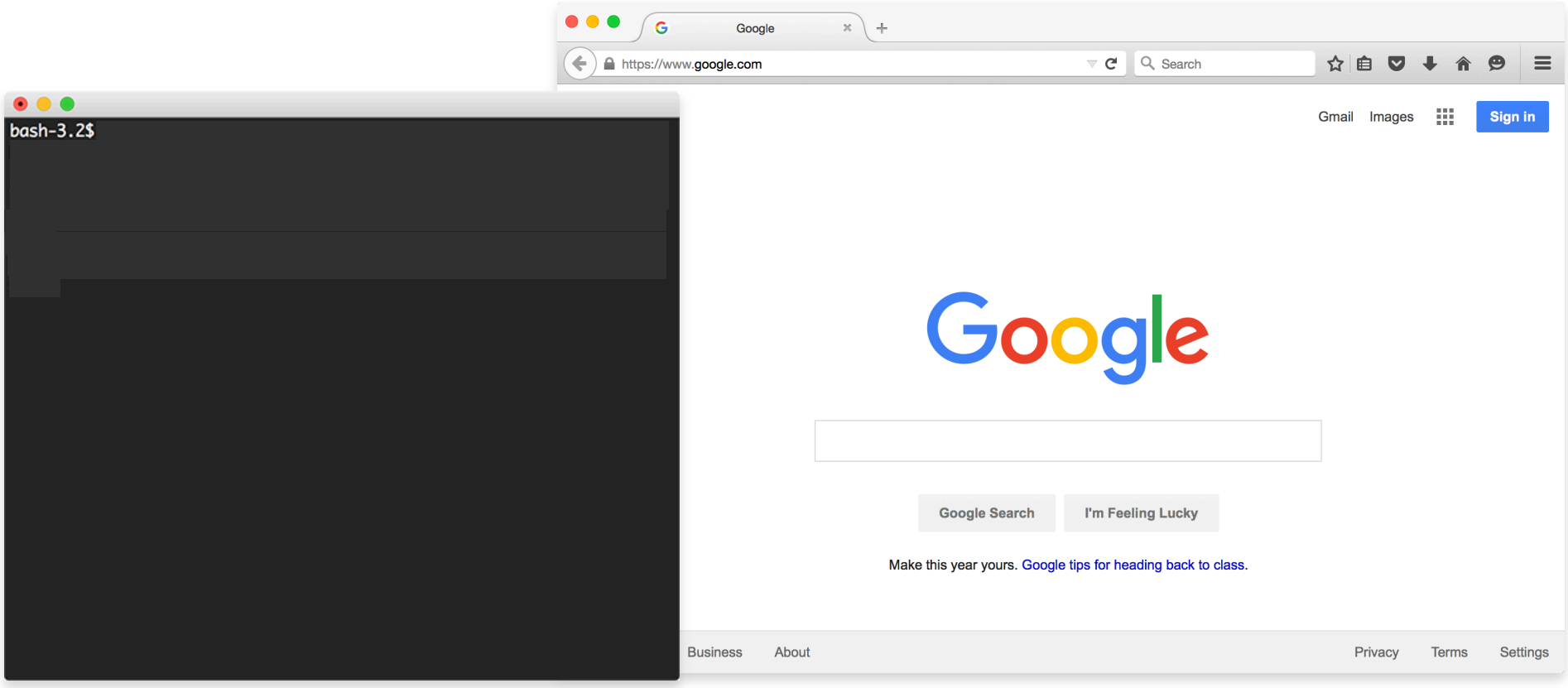
- Python API for Selenium WebDriver
  http://selenium-python.readthedocs.org

# Splunk Synthetic App

- Splunk App for Synthetic Transaction Monitoring
  - http://apps.splunk.com/app/1880
  - By: Elias Haddad

- Setup
  - Install Python if using Universal Forwarder
  - Install selenium and user-agents modules for Python

# Basic Webdriver Demo




# What Data Can We Get?

splunk>

# Sample Data

2015-07-17 10:16:12 app_name="External Sites" transaction_name=”www.google.com" event_type="end" transaction_end="2015-07-17 10:16:12" transaction_end_epoch="1439831772.32" execution_id="a30fd38f-4503-11e5-a699-0050568542ad" transaction_duration=”1.56299996376" browser="Firefox" browser_version="40" os="Windows 7" os_version="" ip="173.194.123.72”

2015-07-17 10:16:02 app_name="External Sites" transaction_name=”www.google.com" event_type="start" transaction_start="2015-07-17 10:16:02" transaction_start_epoch="1439831762.76" execution_id="a30fd38f-4503-11e5-a699-0050568542ad" browser="Firefox" browser_version="40" os="Windows 7" os_version="" ip="173.194.123.72”

# Extracted Fields

- app_name
- transaction_name
- event_type
- transaction_start/end
- Execution_id
- Transaction_duration
- Ip
- Browser
- Browser_version
- Os
- Os_version
- min/max/avg_latency
- Packet_loss

# Application Performance

Edit | More Info

Time Range: Today

Application Name: External Sites

## Current Status

Performance KPI <1m ago

**100.0**

▲ +0%

Availibility KPI <1m ago

**100.0**

— no change

## Performance SLA

<1m ago


## Availibility SLA

<1m ago


## Application Availibility

<1m ago


# Building Tests in Python

splunk>

# Example Manual Test

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import Select
from selenium.common.exceptions import NoSuchElementException
import unittest, time, re
###### STEP 1
### Include the splunktransactions module in your script
from splunktransactions import Transaction

class GoogleTest(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Firefox()

        # replace the above line with the below line to run as Safari
        # Safari driver needs to be downloaded
```

# Set the URL

```
class GoogleTest(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Firefox()

        # replace the above line with the below line to run as Safari
        # Safari driver needs to be downloaded
        #self.driver = webdriver.Safari()
        self.driver.implicitly_wait(30)
        self.base_url = "https://www.google.com/"
        self.verificationErrors = []
        self.accept_next_alert = True

    def test_google(self):
        driver = self.driver

        ###### STEP 2
```

# Name your App and Transaction

```
###### STEP 2
### Provide a name to your Application eg. 'Google'
a=Transaction(driver, 'Google')

###### STEP 3
### assign a name to the transaction by defining a start and an end
a.TransactionStart(driver, 'Google Home Page')
driver.get(self.base_url + "/")
#time.sleep(1)
a.TransactionEnd(driver, 'Google Home Page')
time.sleep(4)

###### STEP 4
### Repeat Step 3 and 4 as needed for as many transactions as needed
a.TransactionStart(driver, 'Search Splunk')
driver.find_element_by_id("gbqfq").clear()
```

# Adding Other Tests

```
    ###### STEP 4
    ### Repeat Step 3 and 4 as needed for as many transactions as needed
    a.TransactionStart(driver, 'Search Splunk')
    driver.find_element_by_id("gbqfq").clear()
    driver.find_element_by_id("gbqfq").send_keys("splunk")
    driver.find_element_by_id("gbqfb").click()
    time.sleep(1)
    driver.find_element_by_xpath("//a[@id='vs0p1']/b[2]").click()
    a.TransactionEnd(driver, 'Search Splunk')
    time.sleep(2)

def tearDown(self):
    self.driver.quit()
    self.assertEqual([], self.verificationErrors)
```


# Problems With The Manual Method

splunk>

# Problems We Encountered

- Two events per test
  - Forced to use TRANSACTION command
  - False positives in app
  - False alerts when service restarted on indexers
- Missing data
  - No server info
  - No context for failures
- Manually building tests is slow
  - Slow to build
  - Difficult to maintain


# Building Dynamic Tests

splunk>

# Goals

- Single event per test
- Capture error information
- Build tests dynamically given an array of info
  - URL
  - App Name
  - Transaction Name
  - Expected title
  - Server

# Single Event Per Test

- Modified **splunktransactions.py**
  - Added Passed / Warning / Failed status
  - Added error info
  - Added server info
  - Combined data into one event

2015-08-17 11:30:47 app_name=”Google" transaction_name=“Home Page" result=Passed duration=1.63 browser="Firefox" browser_version="39" os="Mac OS X" os_version="10.10" ip="173.194.123.72" server="unknown"

# Capturing Failed Test Info

- Failure types
  - Server responds:  Error on page (404, 500, etc)
  - Server doesn't respond:  Problem loading page
  - Timeout issues ( > 30 seconds to load)
    - Check for login prompt
  - Unknown

# Testing for Errors

- Testing for 400 and 500 errors
  - self.assertNotRegexpMatches(page_title, r'[4,5]\d\d', '1’)
- Testing for no page loaded
  - self.assertNotRegexpMatches(page_title, r'problem|not\savailable|error', '2’)
- Testing for a specific title
  - self.assertIn(title, self.driver.title, '3')

# Try / Except

```
def test(self):
    a=Transaction(self.driver, 'Google')
    a.TransactionInfo('Google Home Page')
    try:
        self.driver.get('https://www.google.com')
        page_title = self.driver.title.lower()
        self.assertNotRegexpMatches(page_title, r'[4,5]\d\d')
        self.assertNotRegexpMatches(page_title, r'problem|not\savailable|error')
        self.assertIn('Google Home Page', self.driver.title)
        a.TransactionPass()
    except AssertionError as error:
        # Warn on 401 or 403 errors
        try:
            self.assertNotRegexpMatches(page_title_lc, r'40[1,3]')
            a.TransactionFail(page_title)
        except AssertionError as error:
            a.TransactionWarn(page_title)
        except TimeoutException:
            error = 'Timeout: Page did not load within 30 seconds'
            try:
                alert = False
                while True:
                    Alert(self.driver).dismiss()
                                        alert = True
            except NoAlertPresentException:
                a.TransactionFinish()
                if alert == True:
                    a.TransactionWarn('Test account denied access')
                else:
                    a.TransactionFail(error)
        except:
            a.TransactionFinish()
            error = 'An unknown error occured: %s' % page_title
            a.TransactionFail(error)
            raise
    finally:
        a.TransactionOutput()
```

# Build Tests Dynamically

Set up the array

```
# Array of arrays: app_name, transaction_name, url, title
tests = [
    ['Google','Google Home Page','https://www.google.com','Google'],
    ['Yahoo','Yahoo Home Page','https://www.yahoo.com','Yahoo'],
    ['Bing','Bing Home Page','https://www.bing.com','Bing']
]
```

# Build Tests Dynamically

## Create “Test Builder” Function

```
def test_generator(app_name,transaction_name,url,title):
    def test(self):
        a=Transaction(self.driver, app_name)
        a.TransactionInfo(transaction_name)
        try:
            self.driver.get(url)
            page_title = self.driver.title.lower()
            self.assertNotRegexpMatches(page_title, r'[4,5]\d\d')
            self.assertNotRegexpMatches(page_title, r'problem|not\savailable|error')
            if title != None:
                self.assertIn(title.lower(), page_title_lc)
            a.TransactionPass()
        except AssertionError as error:
            # Warn on 401 or 403 errors
            try:
                self.assertNotRegexpMatches(page_title_lc, r'40[1,3]')
                a.TransactionFail(page_title)
            except AssertionError as error:
                a.TransactionWarn(page_title)
            except TimeoutException:
                error = 'Timeout: Page did not load within 30 seconds'
                try:
                    alert = False
                    while True:
                        Alert(self.driver).dismiss()
                                        alert = True
                except NoAlertPresentException:
                    a.TransactionFinish()
                    if alert == True:
                        a.TransactionWarn('Test account denied access')
                    else:
                        a.TransactionFail(error)
             except:
                a.TransactionFinish()
                error = 'An unknown error occured: %s' % page_title
                a.TransactionFail(error)
                raise
        finally:
            a.TransactionOutput()

return test
```

# Build Tests Dynamically

Add each test to the Unit Test suite

```
count = 0

# Build tests from test array
for test in tests:
    count = count + 1
    test_name = 'test_%03d' % count
    test_case = test_generator(test[0],test[1],test[2],test[3])
    setattr(DynamicTests, test_name, test_case)
```


# Building Dynamic Tests Using Splunk Data

splunk>

# Build Splunk User & Report

- Create a user in Splunk with minimal privileges
- Create a report with that user
  - IIS Logs
  - Top sites by unique users over 7 days

```
index=iis cs_method=GET sc_status=200
| stats dc(c_ip) as count, min(s_port) as port, min(s_ip) as s_ip by cs_host, s_computername
| sort - count
| eval server = upper(s_computername)
| eval category = <insert magic here>
| table category, cs_host, url, server, s_ip
```

# Accessing Report in Python

- Get the splunklib module for Python: http://dev.splunk.com/python

```
import splunklib.client as client
import splunklib.results as results

def splunksearch(offset=0, limit=0):
    HOST = ”localhost"
    PORT = 8089
    USERNAME = ”splunk_user"
    PASSWORD = ”changeme"

    service = client.connect(host=HOST, port=PORT, username=USERNAME, password=PASSWORD, owner="splunk_user", app="splunk-app-synthetic")

    savedsearch = service.saved_searches["SWT_top_sites_by_users"]

    history = savedsearch.history()
    search_kwargs = {
        "offset":   offset,
        "count":    limit
    }
    searchresults = results.ResultsReader(history[0].results(**search_kwargs))
```


# Alerting on Issues

splunk>

# Things to Alert On

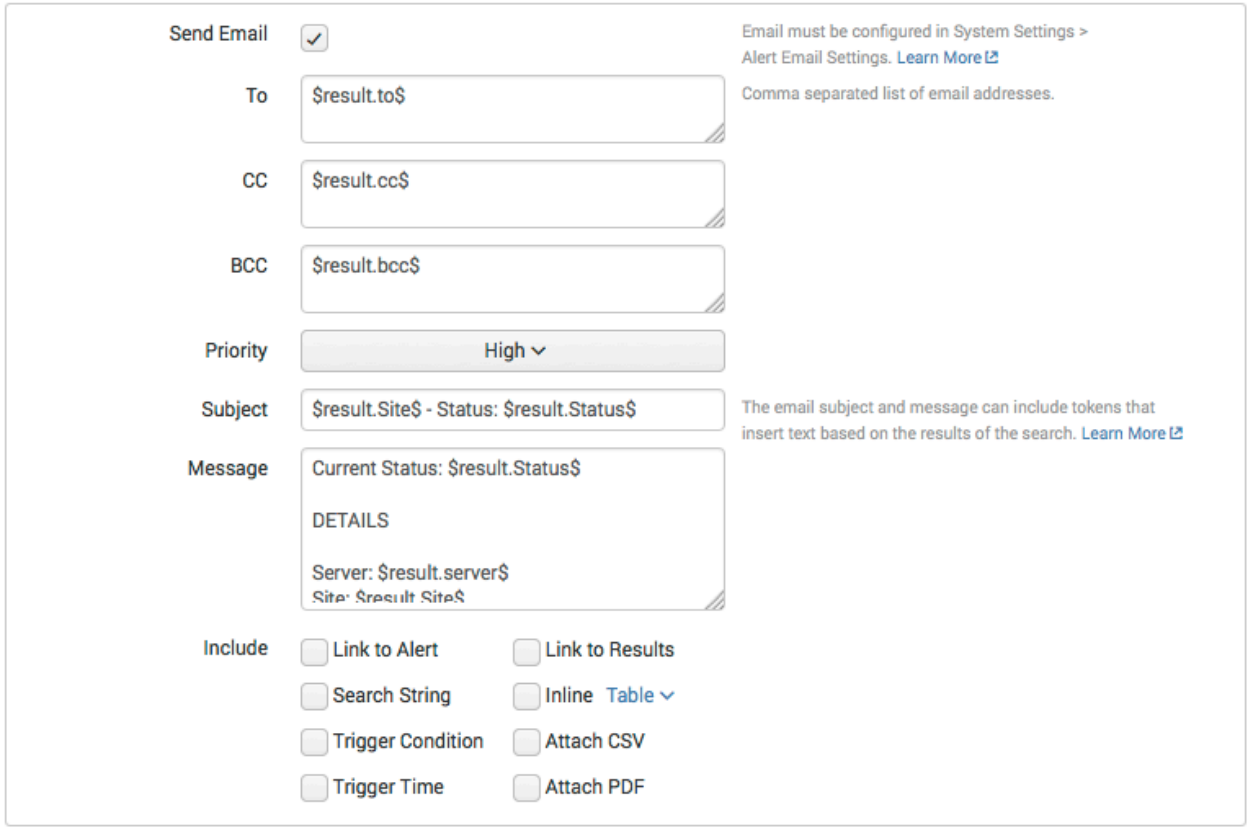
- Synthetic Transactions have stopped
- Test failures on individual websites

# One Query for All Sites

| transaction_name | failure_kpi | to | cc | Bcc |
|---|---|---|---|---|
| Google Home Page | 4 | admin@google.com | | |

```
index=web sourcetype=synthetic:transaction
| transaction transaction_name endswith=result!="Failed" keepevicted=true
| search result=failed eventcount>1
| lookup alert_subscriptions transaction_name output failure_kpi,to,cc,bcc
| fillnull value=2 failure_kpi
| eval Status = if(mvindex(result,-1)="Failed","DOWN","Service Restored")
| search Status=DOWN OR (Status="Service Restored" AND eventcount>2)
| eval Downtime = tostring(duration, "duration")
| fillnull value="justin@pnnl.gov, arzu.gosney@pnnl.gov" to
| eval failures = if(Status="DOWN", eventcount, eventcount-1)
| where failures >= failure_kpi AND _time + duration > relative_time(now(), "-1h")
| rename transaction_name as Site
| eval link = replace(Site, " ", "%20")
| eval eventstart = strftime(_time,"%m/%d/%Y - %I:%M:%S %p")
| eval UniqueId = _time + Status
| eval eventend = strftime(_time + duration, "%m/%d/%Y - %I:%M:%S %p")
| eval details = if(event_type == "end", "Service Restored: " . eventend, "Last Failure: " . eventend)
| table server, Site, Status, eventstart, Downtime, failures, UniqueId, failure_kpi, to, cc, bcc, link, details, error
```

# Configuring the Alert




# Ideas For Expanding

splunk>

# Future Plans

- User interface in app for:
  - Subscribing to alerts
  - Adding sites to manual list
  - “Blacklisting” sites

- Combine test failures inline with IIS logs


# THANK YOU!

Justin Brown

justin@pnnl.gov

@theOtherJustinB

splunk>