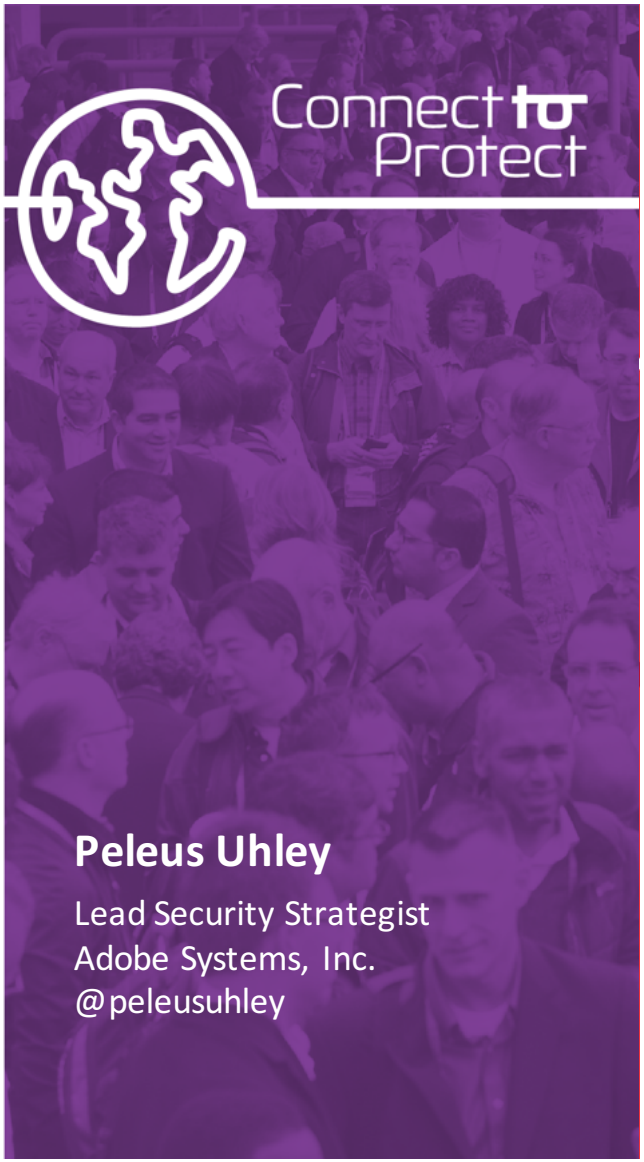
RSAConference2016
San Francisco | February 29 – March 4 | Moscone Center




SESSION ID: STR-F01

# Techniques for Security Scalability

**Peleus Uhley**
Lead Security Strategist
Adobe Systems, Inc.
@peleusuhley

#RSAC

# Agenda

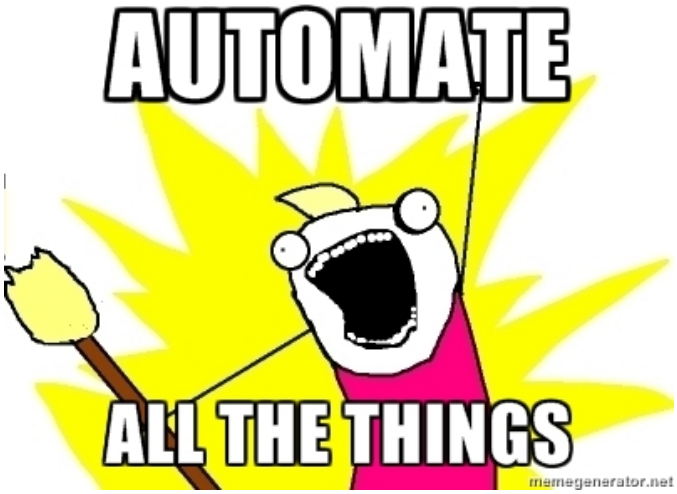

- What is the point?
- Tactical automation (Small scale)
- Automating the automation (Large scale)
- Closing thoughts

# Defining the “why”

# Automate all the things!



- Isn't this what we were doing?
- Tools have been automated in the past. They are called "Enterprise Editions"
- Any idea can be automated
  … even the bad ones...




- http://www.kitchensoap.com/wp-content/uploads/2012/07/automate_all_the_things1.jpeg

# Example: Cloud Automation FTW!



| | Cloud Approach | |
|---|---|---|
| Secure communication | TLS | |
| Authentication | Uname&Pw SessionIDs | |
| Settings Editor | Dynamic GUI in HTML5, 3 JS Frameworks | |
| Scheduler | Fully dynamic timing options | |
| Scalability | Docker, AWS, etc. to scale to an infinite # of sites | |
| Time to implement | 6 mo. to alpha/1 yr to prod | |

Adobe

# Example: The Curmudgeon Pitch


| | Cloud Approach | Crontab Approach |
|---|---|---|
| Secure communication | TLS | SSH |
| Authentication | Uname&Pw SessionIDs | SSH authentication, sudo |
| Settings Editor | Dynamic GUI in HTML5 & 3 JS Frameworks | Vi, Emacs, Pico, SED/AWK, crontab -e |
| Scheduler | Fully dynamic timing options | 30 15 * * * /bin/tool -foo |
| Scalability | Docker, AWS, etc. to scale to an infinite # of sites | We don't have infinite sites. We have 4. It's fine. |
| Time to implement | 6 mo. to alpha/1 yr to prod | Already done. |

Adobe

# A young engineer sees this in a year



# An old engineer sees this in 5 years

# Define the “why” first!



- Automate fingerprinting for tracking your environment?
- Automate to identify something you can actually fix?
- Automate to enumerate a problem that needs funding?
- Automate tedium in order to free up resources?
- General metrics on security health?

Adobe

# Data affects management decisions


- Collecting the point-in-time assessment requires more work before the first successful feedback on the project
- How the “scale” data is presented skews their perception of the “large” problems
- The story that the data tells can affect funding
- Large scale data that is not evenly collected can make issues appear disproportional in relation to each other

# Traditional "How" Approach: The security assessment


- Run a WAPT against the entire web site to find everything
- A good option if you have time for manual follow up
- Bad for statistics, lots of false positives
- Accuracy requires time for tuning & snowflakes
- Action items can be unclear or open-ended

Adobe

# Alternative Approach: Security assertions



- A true/false test for a specific, defined security property
- Easy to understand test case for developers
- Good for statistics
- Easy to maintain
- Points the team to specific and actionable work
- Requires effort in order to get a bundle of tests

Adobe

# Example: Solving XSS through automation



- Security Assessment: How many XSS bugs do you have today?
  - A point-in-time measurement
  - Leads to whack-a-mole response to the data collected
  - Difficult to collect due to tuning, sophistication of the tool, etc.

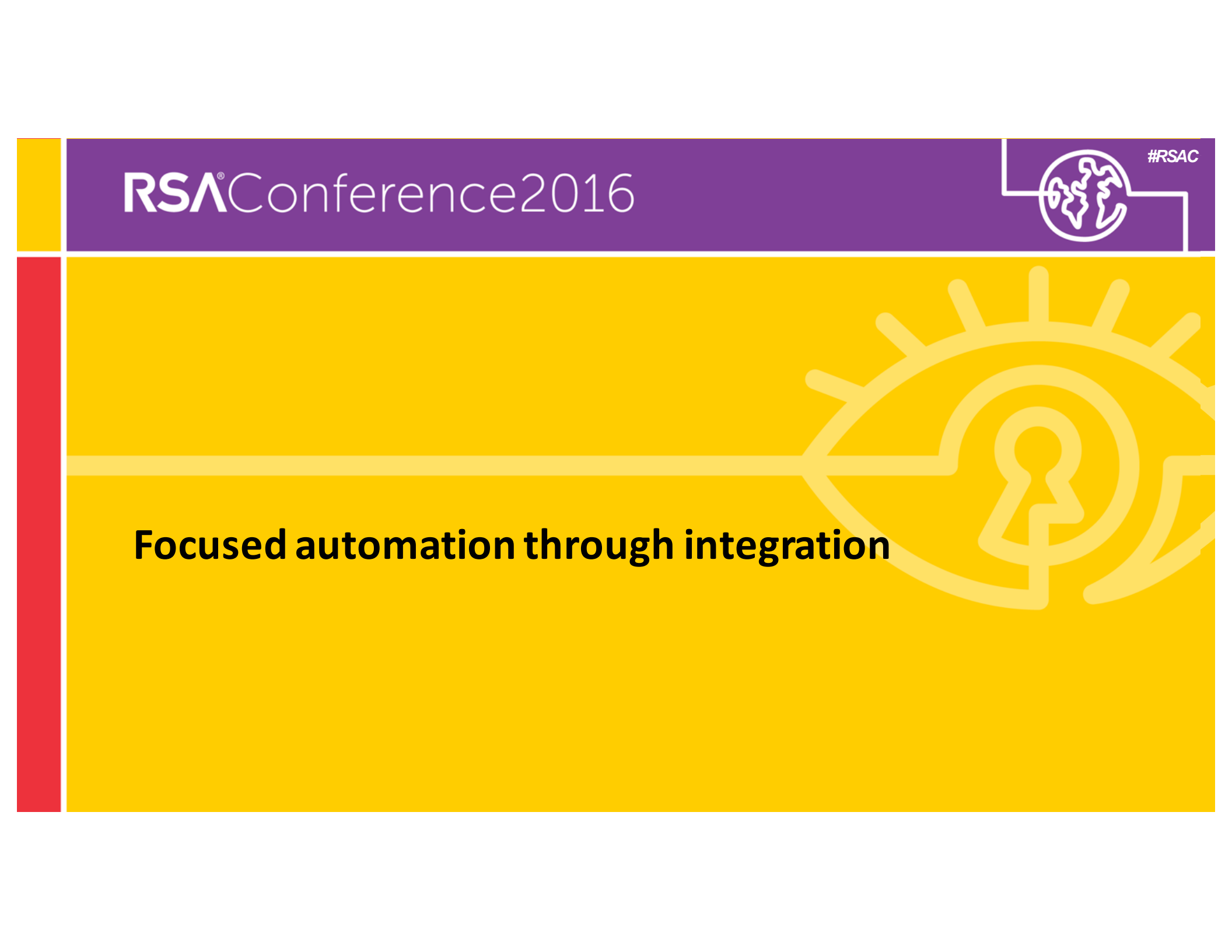
- Security Assertion: Which projects return a CSP header?
  - Team focuses on the CSP feature roadmap
  - The team is focused on deploying mitigations instead of individual bugs
  - Trivial to measure with high accuracy

Adobe

# You may need both


- If you are in a responsive state, you may need the point-in-time assessment.
- If you are in a proactive state, you may have the luxury of measuring your goals.
- You should ask the question of which is more important before diving into the implementation details.

Adobe



# Focused automation through integration

# Integration into existing systems



- Build automation – Chef, Puppet, etc.
- Source repositories – Git, Perforce, etc.
- Build tools – Maven, Jenkins, etc.

CHEF™

GitHub

Maven™

Adobe

# Example projects



- GIT commit checks for PCI compliance
- Static analysis integrated into code quality tools
- Instance trackers that record AMI versions in use
- Maven integration for enumerating vulnerable 3rd-party libraries

Adobe

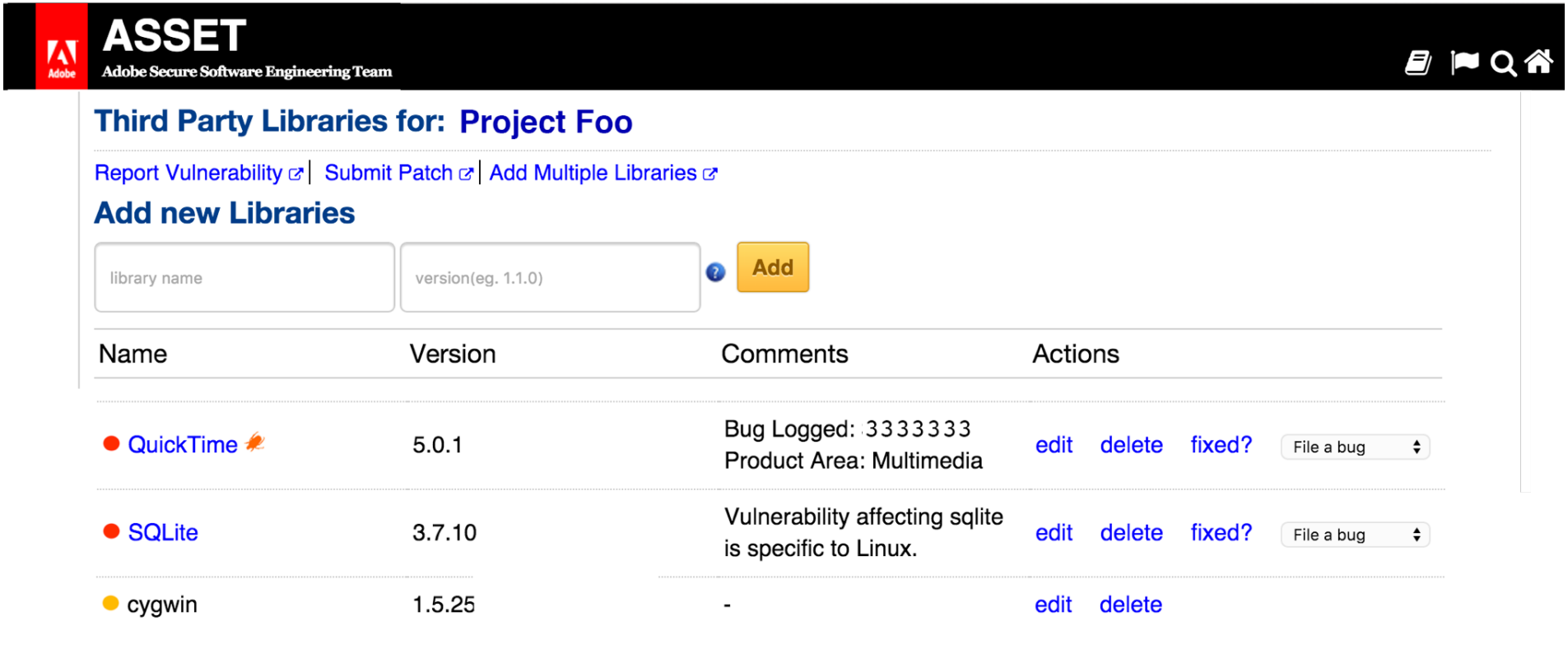
# Third-party library & component tracker


- Processes feed of 3rd-party library security updates
- Tracks internally built shared libraries
- Maintains dependency trees between the two
- Highlights 3rd-party library issues with entries in ExploitDB
- Emails owners when new issues arise
- API to allow nightly updates from Maven-like build tools

Adobe

# 3rd – party data for a demo environment





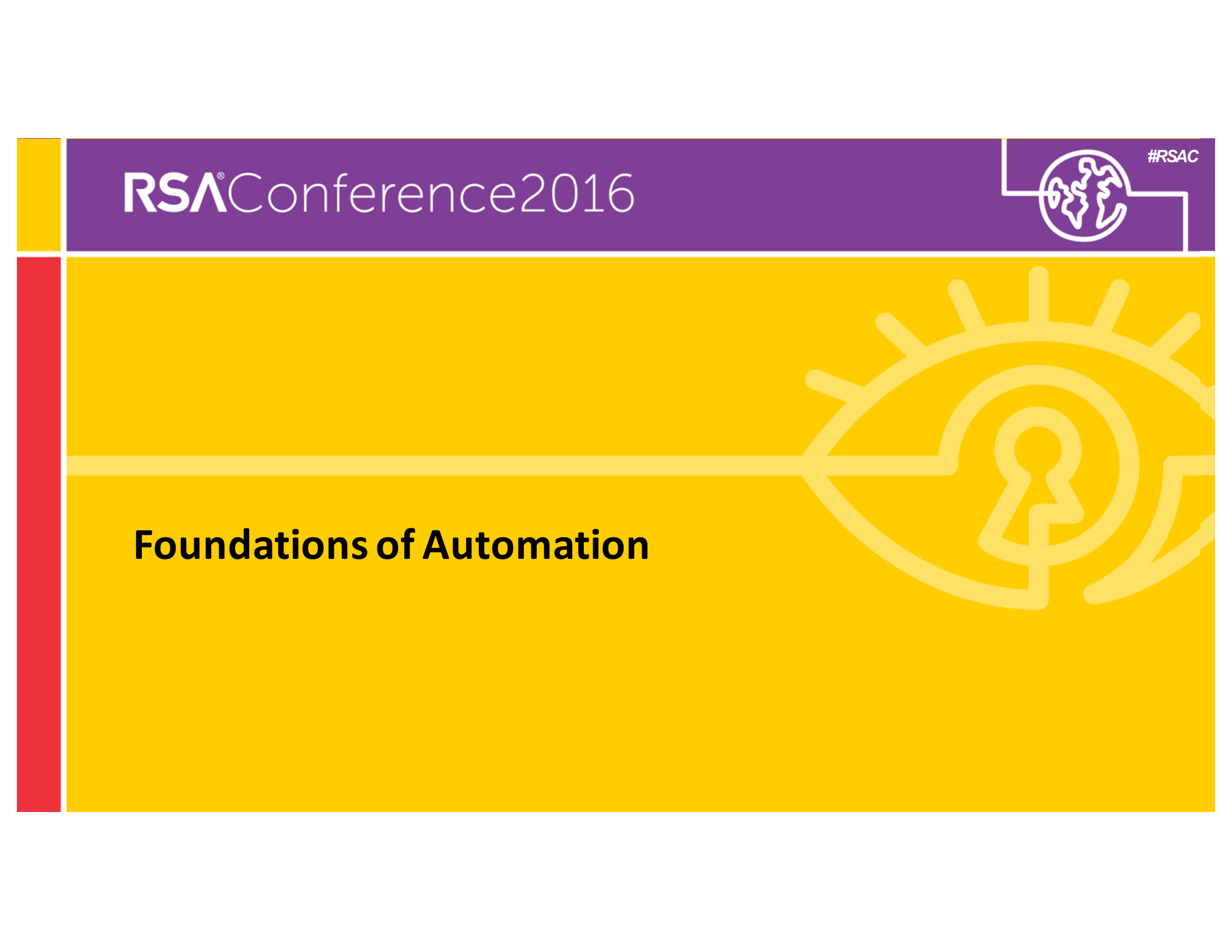
# GIT code review checker


- Jenkins plugin
- All commits belong to pull request
- The committer is not the creator of the pull request
- Enables code-review compliance (PCI, et al)

Adobe

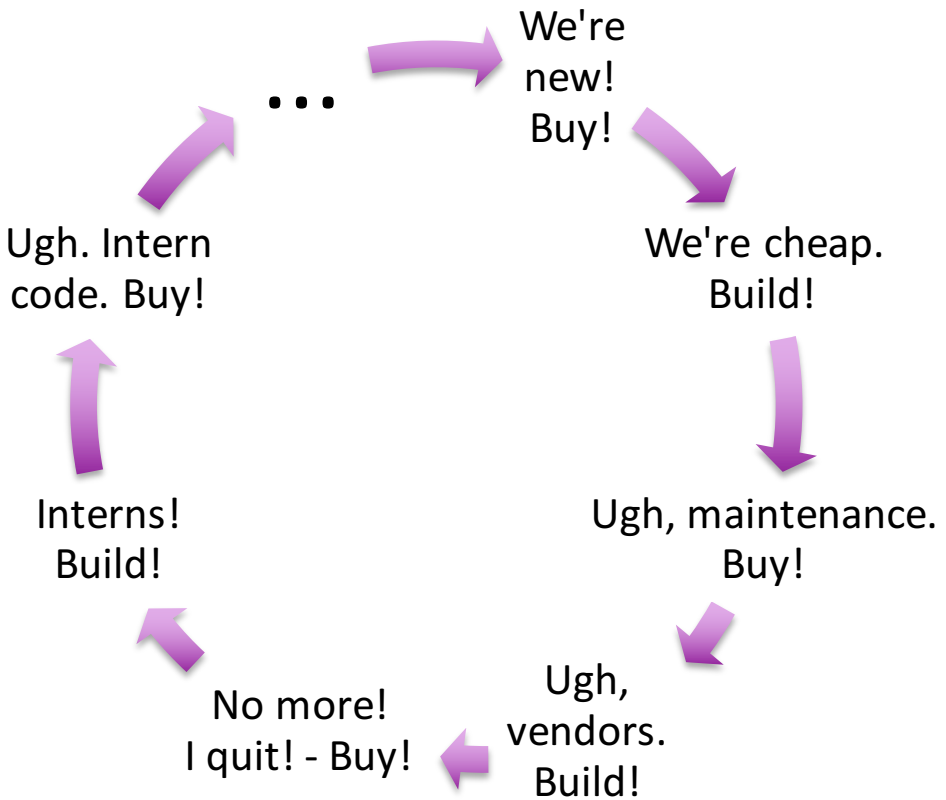
# Advantages of inline models


- Specific – A singular, well defined unit test
- Measurable – Pass/Fail results
- Actionable – It is clear how to handle a Fail result
- Relevant – Required for compliance or similar need. Based on tools that the team already uses.
- Time-Bound – These projects were done as “Brown Belt” projects in our security training program



# Foundations of Automation

# The Build vs Buy cycle



# Stealing ideas and tools (borrow)


- Baseline frameworks (Gauntlt, Mittn, etc.)
- Automation systems (Mozilla Minion, Pitke's SAF, etc.)
- Full service (Salesforce Chimera, Twitter SADB, etc.)
- Tool consolidators (ThreadFix)

Adobe

# Behavior Driven Development Tools


| | Mittn | Gauntlt | BDD-Security |
|---|---|---|---|
| Primary Language | Python | Ruby | Java |
| BDD Framework | Behave | Cucumber | jbehave |
| Default Web App Pen Test Tools | Burp Suite, radamsa | Garmr, arachni, dirb, sqlmap, curl | Zap, Burp Suite |
| Default SSL analysis | sslyze | heartbleed, sslyze | TestSSL |
| Default Network Layer Tools | N/A | nmap | nessus |
| Windows or Unix | Unix | Unix** | Both |

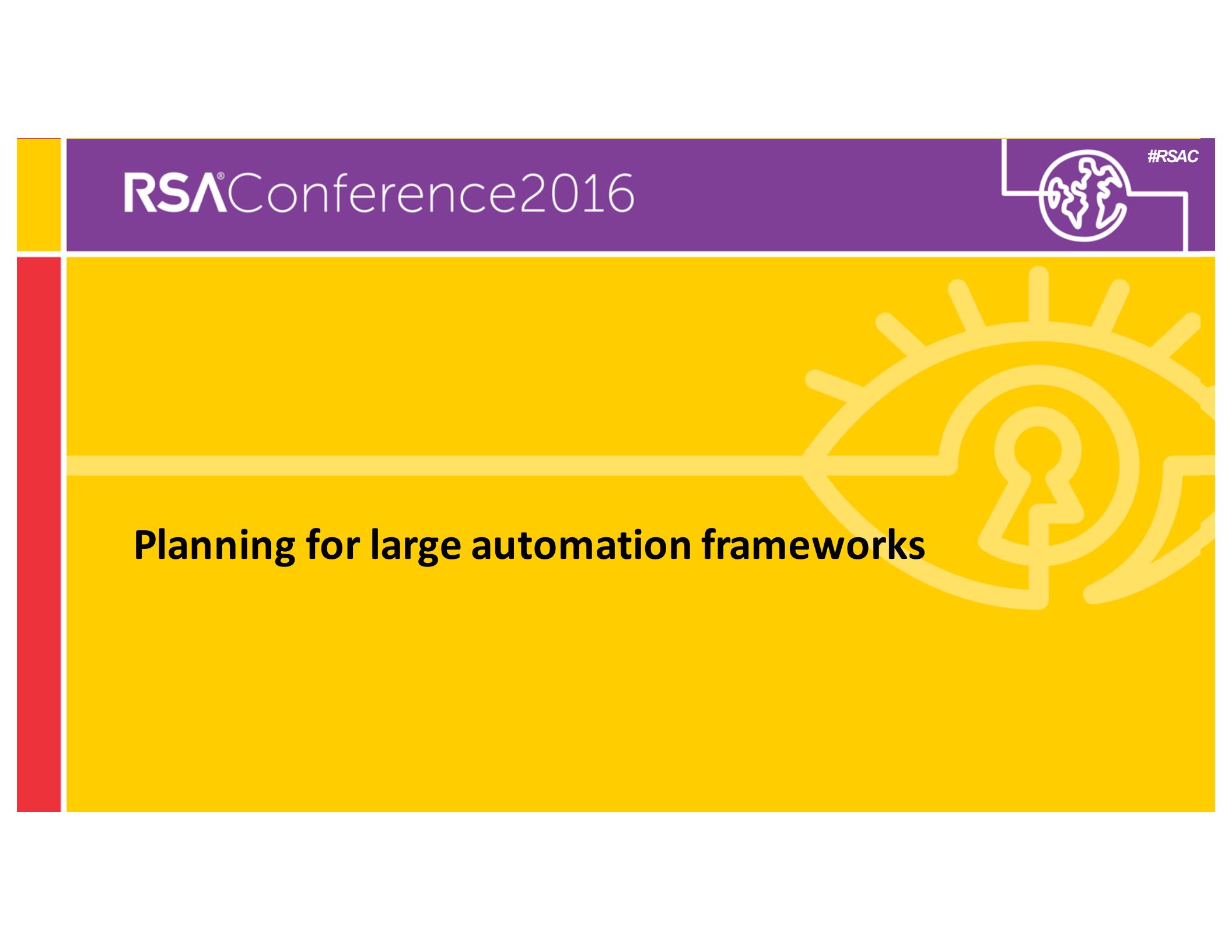
# Gherkin security assertion


*Scenario: Lock the user account out after 4 incorrect authentication attempts*
*Meta: @id auth_lockout*
*Given the default username from: users.table*
*And an incorrect password*
*And the user logs in from a fresh login page 4 times*
*When the default password is used from: users.table*
*And the user logs in from a fresh login page*
*Then the user is not logged in*

# Can you maintain it after launch?



- Environment patches?
- Configuration updates?
- Tool updates?



# Planning for large automation frameworks

# Development estimates



- Building a fully scalable architecture requires 6 months to a year
- Do you need incremental delivery for management?
- Do you need incremental delivery to adjust the design as you go?
- Scan from the internal or external network?

# Separate tools from the core engine


- Tools need to scale to multiple machine instances to meet demand
- Dependency conflicts will arise from shoving too many tools on one instance
- Separate instances will make queueing easier

Adobe

# Separate reporting from data aggregation


- Common report requirements:
  - Excruciating detail for engineers
  - High level “30,000 ft” pictures for managers
  - Selective views for compliance
  - Special snowflakes
- A REST API into the database allows people to pull what they need

Adobe

# Plan for dynamic variables

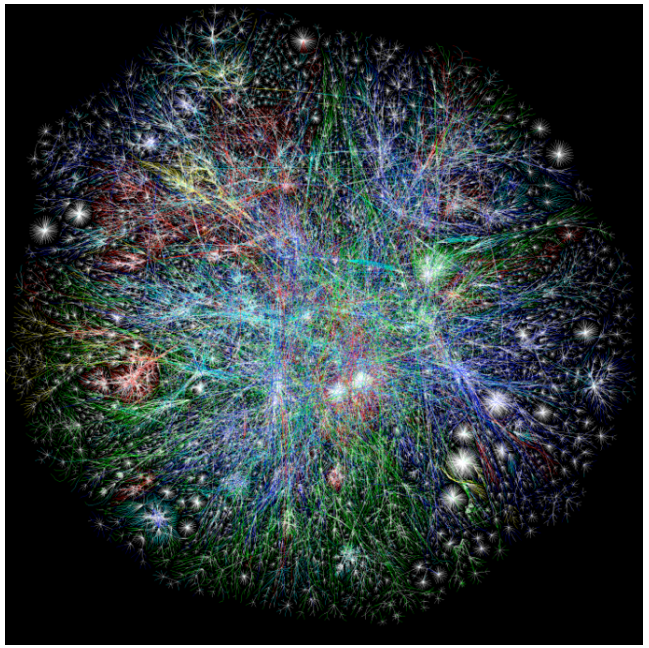

- Targets will vary across dev, stage and prod
  - Different IPs/Hosts
  - Different authentication credentials
  - Different scan rules
- Sharing across orgs will mean custom configs per tool client
- How many types of inputs will you need to support?

Adobe

# Have a picture of your organization



- Automation built just for tracking and updating this data
- Potential sources
  - Internal sources: DNS servers, AWS interface, human entry, etc.
  - Open Source: Scans.io, manual research (whois, nslookup, etc.)
  - Commercial: RiskIQ, Farsight, etc.



http://www3.nd.edu/~networks/Image%20Gallery/Large%20Images/Opte%20Project_lg.png

Adobe

# Review tool APIs before choosing


- ZAP is great but it isn't well documented for automation purposes (yet).
- Can you customize the tool's tests? Can you run a single test?
- Can the tool authenticate to your environment?
- Tool pre-requisites (Java, etc.)?
- Does it listen on a port which may limit 1 tool per instance?

Adobe

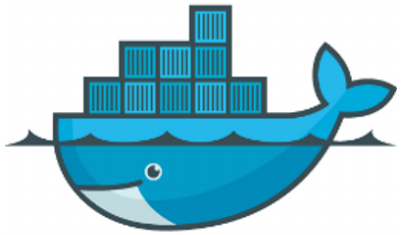
# REST API scale challenges


- You can’t spider a REST API
- What format for providing the baseline to the environment?
- Once you have the baseline, how do you update it over time?
- JSON response parsing

# Platform choices



- AWS/Azure/Compute?
  - {Insert lengthy technology debate here}
- Docker-izing (or favorite alternative)
  - Great for prototyping on your desktop*
  - Docker support in AWS



amazon web services™

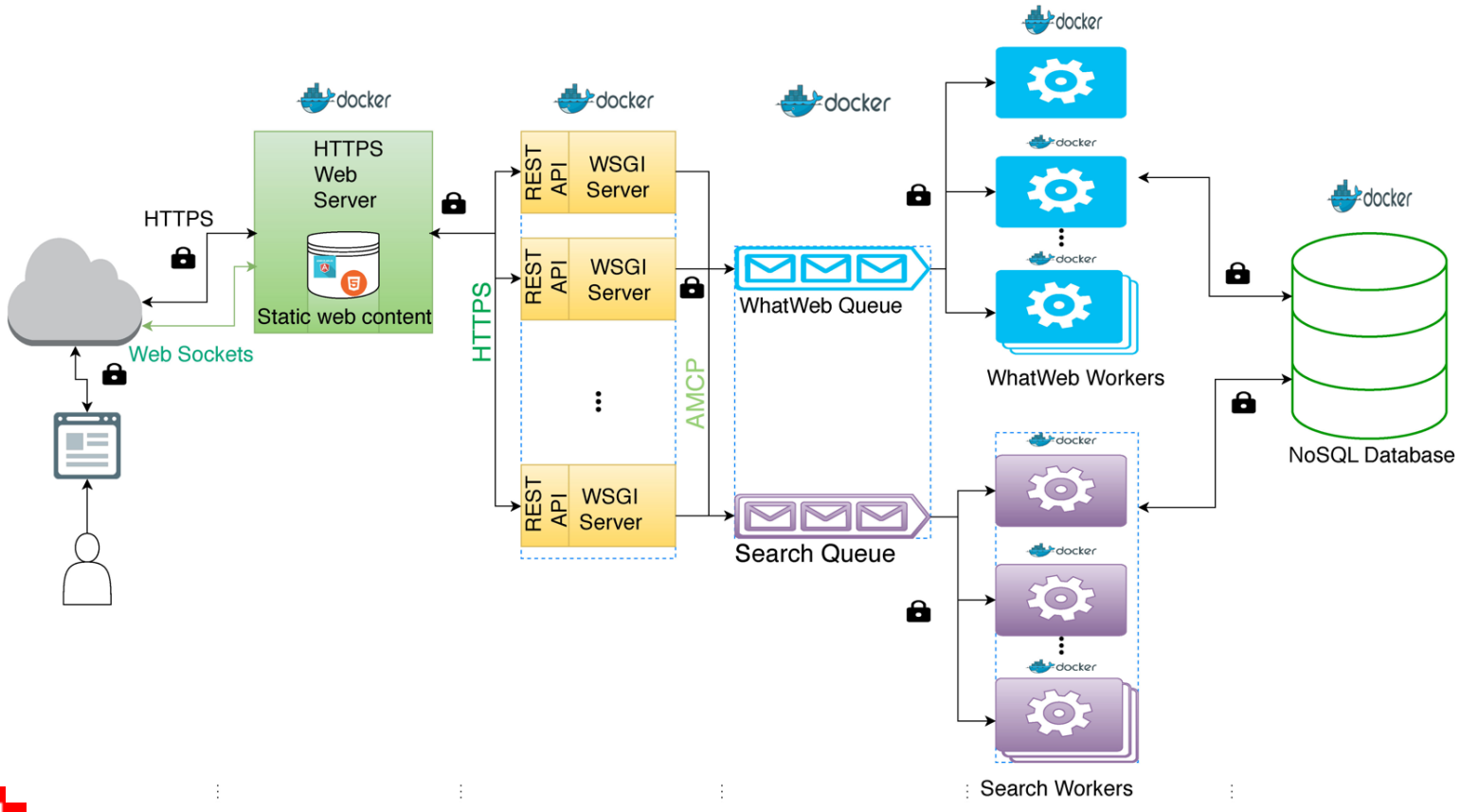
Windows Azure™

Google Compute Engine

Adobe

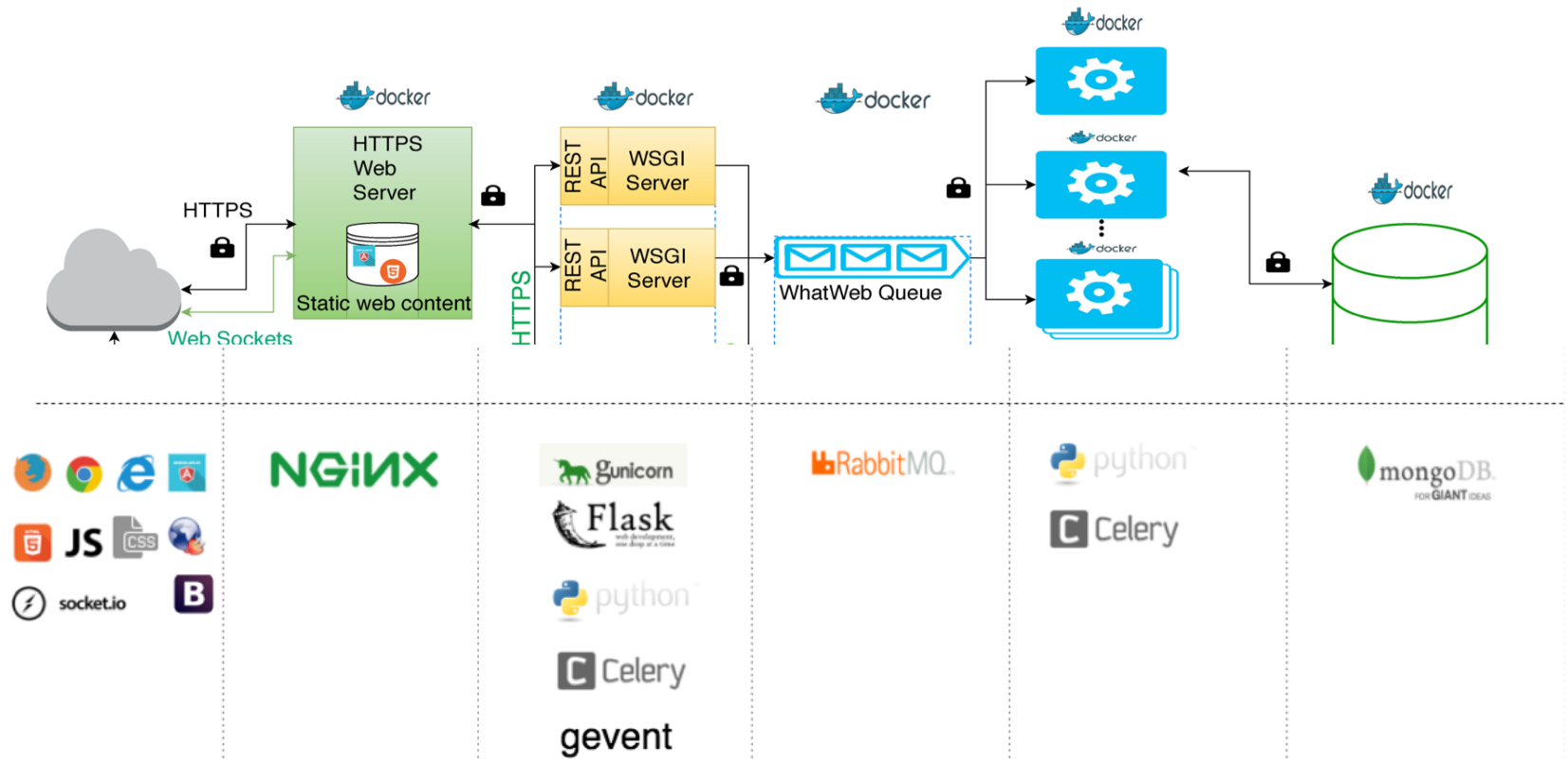
# Example 2: WhatWebber



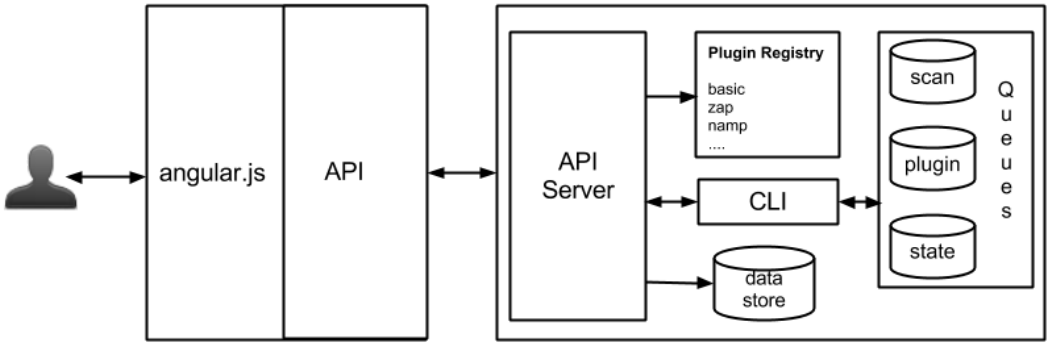


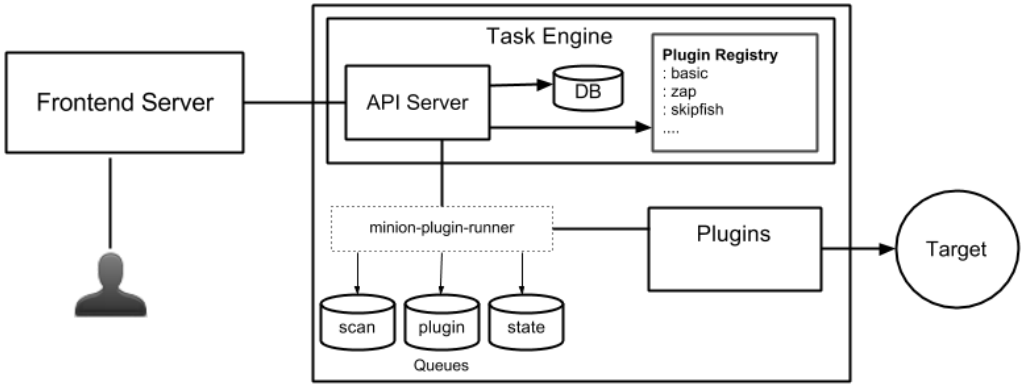
# Example 2: WhatWebber





# Example 1: Mozilla's Minion




https://wiki.mozilla.org/images/8/86/Minion-03-diagram-draft.png

# Example 3: Security Automation Framework


- Security assertion based
- Wrapper to open-source Gauntlt framework
- Tool agnostic and team agnostic
- Authoring of tests can be done by anyone
- API design allows it to be a hub rather than a destination

Adobe

# SAF prototype environment



## Security Automation Framework

HOME | INPUTS | SCAN | ASSERTION | ASSERTION REQUEST | DASHBOARDS | Welcome puhley!

### Assertion details

| Assertion name | Description | Assertion Filename | Assertion Output type | Success message | Fail Message | Script file name | Features | | |
|---|---|---|---|---|---|---|---|---|---|
| SSL Labs scan | Scans the hosts using SSL labs api | ssllabs-scan.attack | json | SSL Labs grade higher or equal to B- | SSL Labs grade lower than B- | | View features | Edit | Delete |
| SSLv3 support | Checks for SSLv3 support | check_sslv3.attack | text | SSLv3 not supported | SSLv3 supported | | View features | Edit | Delete |
| Server Connectivity | Checks if server can be reached | check_server_up.attack | text | Yes | No | | View features | Edit | Delete |
| SHA1 certificates support | Checks for SHA1 certificates support | check_chrome_sha1.attack | text | SHA1 certificates not supported | SHA1 certificates supported | | View features | Edit | Delete |
| Heartbleed Vulnerability | Checks for Heartbleed Vulnerability | heartbleed.attack | text | Not Vulnerable to Heartbleed | Possible Heartbleed Vulnerability | | View features | Edit | Delete |
| Check X-Frame-Options header | Checks for clickjacking vulnerability | check_xframe_headers_options.attack | text | X-Frame-Options set to SAMEORIGIN | Possible clickjacking Vulnerability | | View features | Edit | Delete |

Adobe

# Build towards automation in small steps


- A “loose coupling” design means that you can deploy components as you go.
- Sub-components can include: The tool runner, fingerprinting, scalable Docker images, etc.
- Always ask yourself, “Why not cron?,” to keep focused on the value add of the project.

# Conclusion

# Conclusion



- Know what you will do with the output before you build.
- Not all security automation has to be “at scale”
- Study ”prior art” before tackling a large project

# “Apply” Slide



- Next week you should:
  - Identify business needs which could be supported by automation
- In the first three months following this you should:
  - Identify where security automation can be integrated into existing tools
  - Create a roadmap to incrementally build towards a large scale framework
- Within six months you should:
  - Start coding your first automation components
  - Use large scale data to make better security decisions

Adobe

# Contact:


- https://twitter.com/peleusuhley
- https://blogs.adobe.com/security/
- puhley@adobe.com