

# AUTO-AUTH-SERVER

---

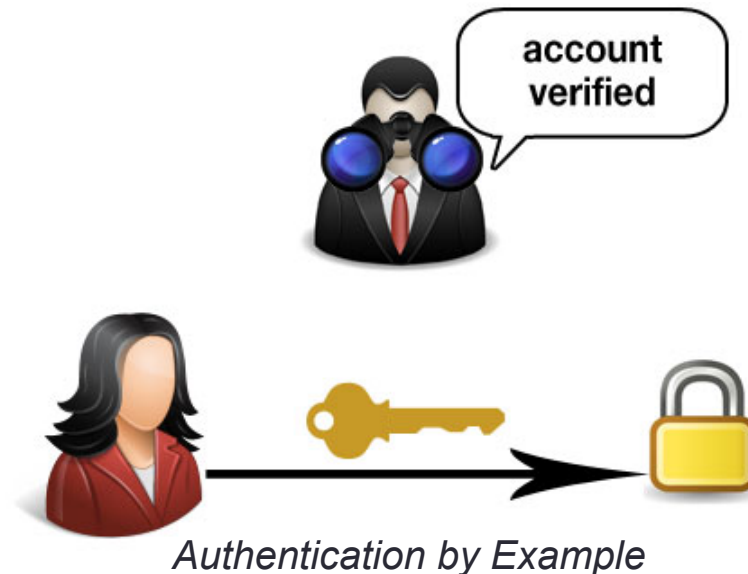
A Masters Project on Authentication by Jay Waldron  
Under the Direction of Professor Ron Cytron

# AUTHENTICATION

---

# Authentication & Account Membership

- When should authentication be required?
  - WebSTAC
  - Campus Circulator App
- How do I know you are who you say you are?
  - Authentication by example
    - Bank of America – enter your PIN before I will help you
  - Can we do this from a program? Automatically?



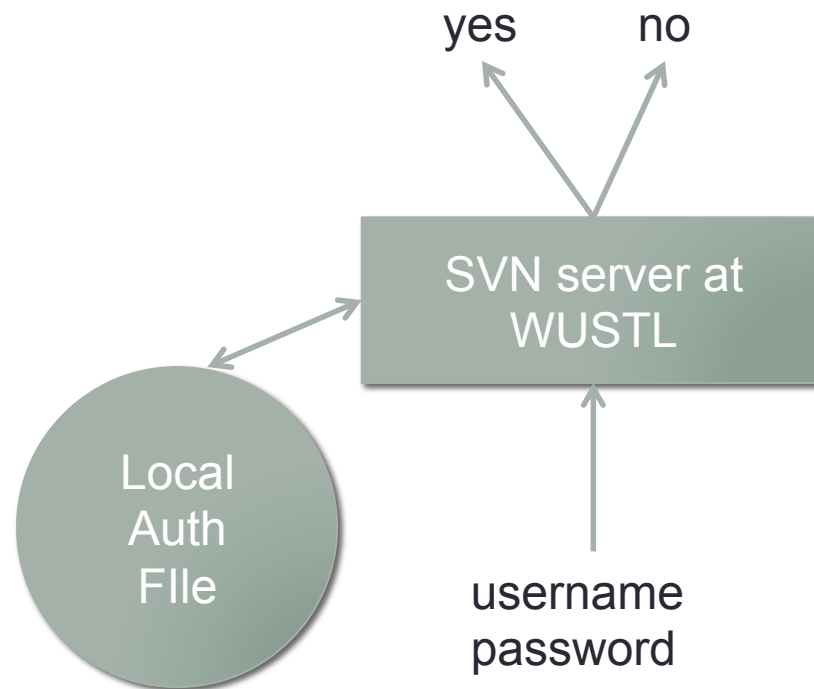
# Circumventing Closed Portals

- How can we authenticate at a closed portal?
  - Google vs. WUSTL
  - Argument: Anything accessible via a web browser, we should be able to automate within reason.
  - Campus Circulator App
    - WUSTL's portal is closed, but we can always authenticate over the web
  - How can the App “watch” you authenticate?
    - It can ask you for credentials and then simulate you logging in
    - How can the program determine if the simulation actually logged you in
- In summary:
  - 1) An App that wants to authenticate using...
  - 2) A closed authenticator
  - This problem was the primary focus of the project

# THE PROBLEM & PROJECT GOALS

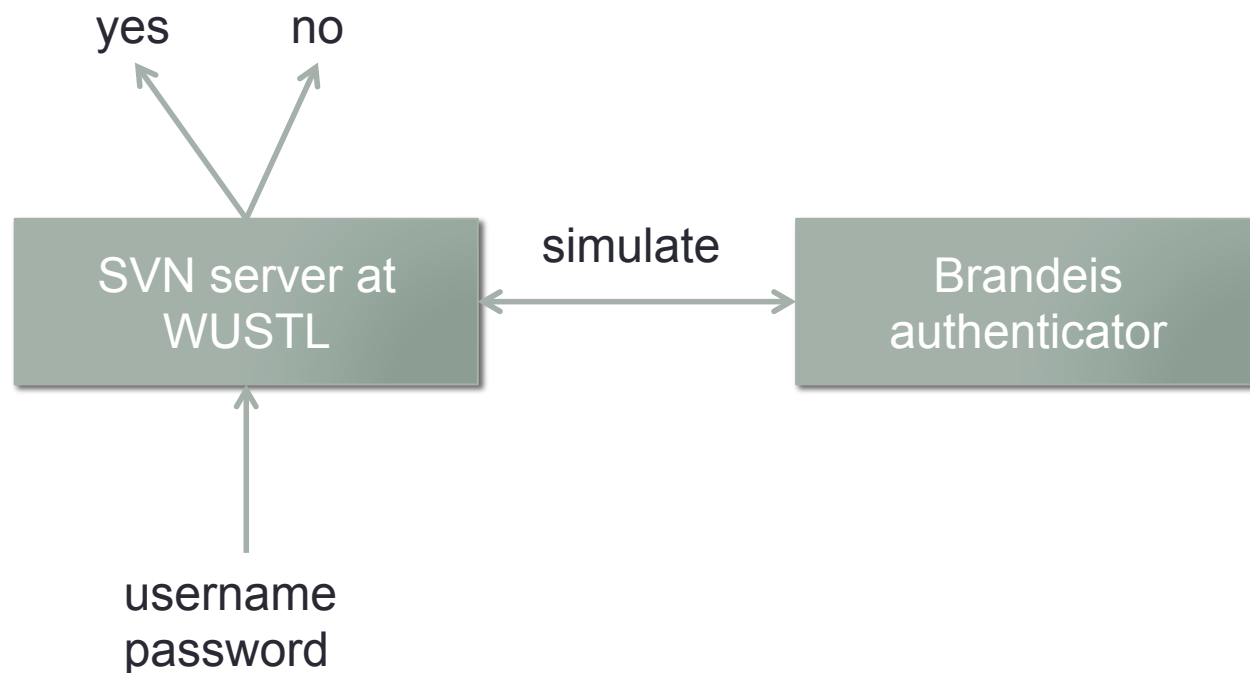
---

# Overview of Problem



We have to manage this!

# Overview of Problem



How can we tell?

# How can we tell?

- Focus: Automated login exclusively for Consortium Schools
- Initial login success determined with key phrase detection:
  - WUSTL: we know everything
    - Watch for success / failure phrases & return as thus
  - Other schools: we know only one kind: incorrect page
    - Can't generate successful logins or lockout-type failures
    - Only finding failure phrases, otherwise call it a successful login
    - Very weak indicator & problematic
      - Altered web design!
      - Locked out!
      - Change your password!



# Exploring the No-Account Problem

- Goal: Determine success of login on a page we've never seen before
- Ideas (from bad to good)
  - Mechanical Turk: concerns over sending potentially sensitive info
  - Phrase Detection: problematic & locked to predetermined sites
  - Cookies: the de-facto standard for session variable storage today
    - Number of cookies is a great indicator
    - Number of cookies alone is not enough (WUSTL cookies: 5 vs 8 vs 6)

# Expanding to the General Case

- Can we do this more generally?
  - Outside of just predefined Consortium School Sites
  - Let the system admin decide what site they want to open auth to
- For better reliability and generality, need a better page-type decision mechanism
  - Explore the problem from a ML standpoint

# TAKING A DATA MINING APPROACH

---

# Existing Research & Tools Used

- Existing research on classifying web pages
  - Primarily in relation to type (blog, media, news, etc.)
  - Primarily in relation to algorithmic goals similar to Page Rank algo
  - Not much on classifying auth responses or pos/neg pages
- WEKA
  - Entire system built in Java (covered in-depth later)
    - Wanted a native Java solution
  - WEKA ML library was used due to
    - ease of implementation
    - good reputation
    - pure Java construction

# Feature Design

- Inspiration taken from *Chau (2007)*, *Hertzog (2013)*
- Classification occurs on three feature types: Text, Content, Structure
  - Adapted these categories when picking features
  - Experimented with handful of features, with the below being finalized:

## • Text

- Num words on page / in title
- Length of URL
- Num cap words
- Existence of keyword variations:
  - Login / Logout
  - Create Account
  - Password / Username
  - Forgot
  - Reset
  - Locked
  - Contact Admin

## • Content

- Number of:
  - Divs
  - Frames
  - Forms
  - Input Fields
  - Images
  - Links
  - Distinct Colors
  - Distinct Text Sizes
  - Distinct Font Families
  - Words on page
- Boolean is displaying red text
- Total Image pixel area

## • Structure

- Num cookies
- Num secure cookies
- Num Distinct Cookie Source Domains
- Num Links

# Training Data Sites

- Obtained accounts for myself on the following sites:
  - Successful Login Vectors
    - Amazon
    - Google
    - Evernote
    - Reddit
    - Github
    - BodyBuilding
    - Coderanch
    - Glassdoor
    - LinkedIn
    - Facebook
    - eHarmony
    - CollegeProwler
    - Paypal
    - Payscale
    - Groupon
    - Shutterstock
    - Spotify
    - Yahoo
    - Meetup
    - Sparknotes
    - Wikipedia
    - Stumbleupon
    - Internet Archive
    - Engadget
    - Fidelity
  - Unsuccessful Login Vectors
    - All of the successful login sites, plus
      - Kayak
      - Namecheap
      - Newegg
      - Steam
      - Microsoft
      - Citicards
  - Lockout signals were also generated as false samples where applicable

# Selecting a Learning Approach

- Wanted the following:
  - A fast, robust, and somewhat forgiving learning approach
  - Reasonably understandable for a less-versed server admin
    - Preferred “white-box” method so an admin can identify failure
- Initial approach: Decision Tree due to the above
  - Used WEKA’s implementation of C4.5 algorithm (called a J48 tree)
  - No Pruning Performed
  - WEKA makes switching out models relatively easy
- Also tested Random Forest model for comparison
  - 1000 trees

# Experiment Design

- Goals:
  - Which test approach provides best the best accuracy?
  - Which test approach provides should be pushed to production?
- Tests:
  - Four general test approaches were used
  - Using LOOCV on all experiments
    - goal to compare approaches to each other

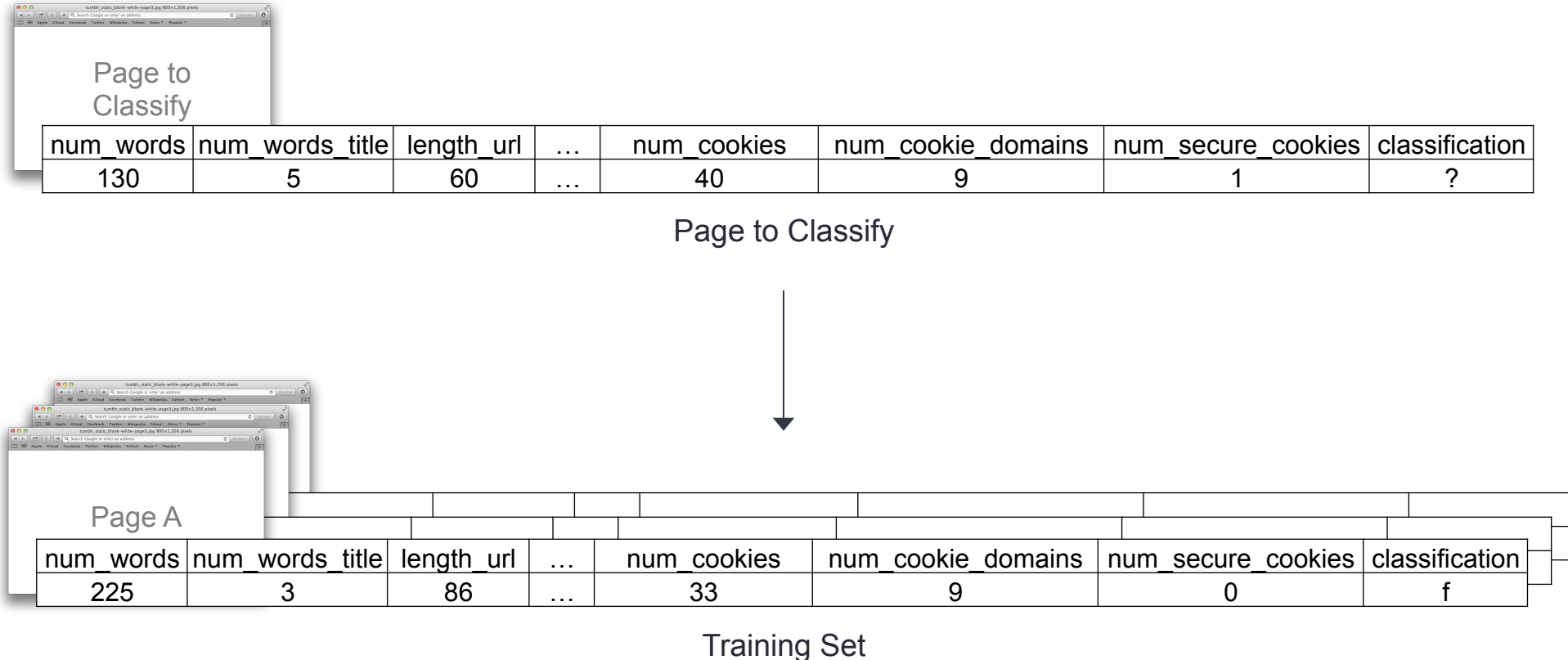
	Single Classifier	Triple-Classifier
Raw Features	Approach 1	3
Baseline Subtracted Features	2	4

Approaches detailed over next few slides



# Experiment Design: Approach 1

- Throw all raw page features into one classifier



# Experiment Design: Approach 2

- Create a guaranteed failure page when server starts
  - Specialized for each site
  - This becomes a baseline
  - Future pages have this baseline subtracted from their features

Baseline for  
Site A

num_words	num_words_title	length_url	...	num_cookies	num_cookie_domains	num_secure_cookies	classification
225	3	86	...	33	9	0	f

Definite Failure (Collected baseline for a given site)

Page to Classify  
for Site A

num_words	num_words_title	length_url	...	num_cookies	num_cookie_domains	num_secure_cookies	classification
130	5	60	...	40	9	1	?

num_words	num_words_title	length_url	...	num_cookies	num_cookie_domains	num_secure_cookies	classification
225	3	86	...	33	9	0	f

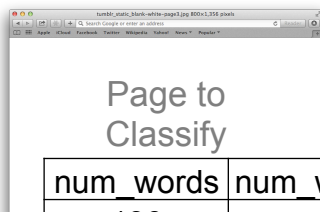
=

num_words	num_words_title	length_url	...	num_cookies	num_cookie_domains	num_secure_cookies	classification
-95	2	-26	...	7	0	1	?

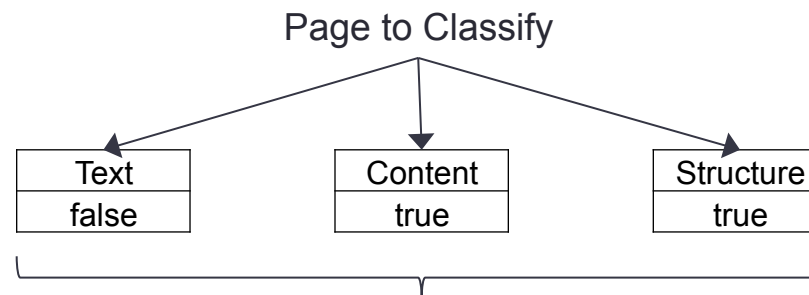
Page to Classify Features

# Experiment Design: Approach 3

- Separate classifier for each of the three feature categories
  - text, structure, content
  - Using raw data feature design from Approach (1)
- Combine these classifiers
  - Return the greater classification of the three



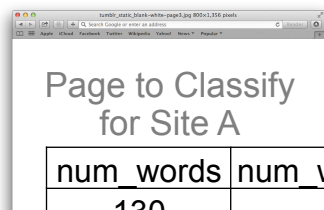
num_words	num_words_title	length_url	...	num_cookies	num_cookie_domains	num_secure_cookies	classification
130	5	60	...	40	9	1	?



Classified as "true"

# Experiment Design: Approach 4

- Same as Approach (3)
  - Uses baseline subtracted feature design from Approach (2)

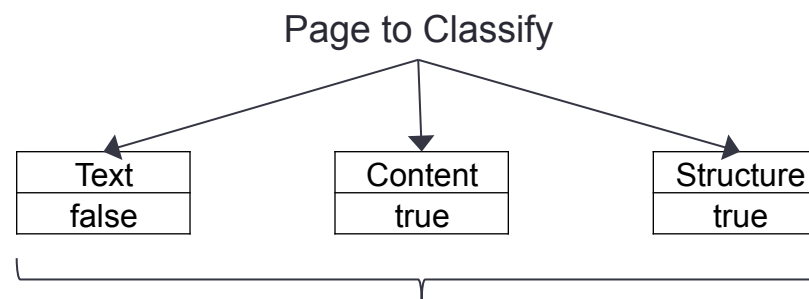


num_words	num_words_title	length_url	...	num_cookies	num_cookie_domains	num_secure_cookies	classification
130	5	60	...	40	9	1	?

num_words	num_words_title	length_url	...	num_cookies	num_cookie_domains	num_secure_cookies	classification
225	3	86	...	33	9	0	f

=

num_words	num_words_title	length_url	...	num_cookies	num_cookie_domains	num_secure_cookies	classification
-95	2	-26	...	7	0	1	?



Classified as "true"

# Test Results

- Three preprocessing methods examined
  - Tested all 4 approaches on each of these preprocessing methods
    - With both Decision Tree & Random Forest
- A. Removed duplicates
- B. Method A + normalized data from 0 to 1
- C. Method B + used a greedy feature selection algorithm

## A) Removed Duplicates

	DT Accuracy	RF Accuracy	Sample Size
1: 1-classifier raw	93.2%	94.3%	88
<b>2: 1-classifier subtracted</b>	90.7%	<b>97.7%</b>	43
3: triple-classifier raw			
text	93.8%	90.1%	81
content	78.7%	81.3%	75
structure	62.2%	71.6%	74
<b>combined</b>	<b>94.3%</b>	93.2%	43
4: triple-classifier subtracted			
text	88.4%	95.4%	43
content	84.2%	94.7%	38
structure	77.1%	91.4%	35
combined	90.7%	95.4%	43

## B) Removed Dups, Normalized [0,1]

1: 1-classifier raw	93.2%	93.2%
<b>2: 1-classifier subtracted</b>	90.7%	<b>97.7%</b>
3: triple-classifier raw		
text	93.8%	90.1%
content	78.7%	78.7%
structure	62.2%	70.3%
<b>combined</b>	<b>94.3%</b>	92.1%
4: triple-classifier subtracted		
text	88.4%	93.0%
content	84.2%	94.7%
structure	77.1%	94.3%
combined	90.7%	95.4%

## C) Removed Dups, Normalized, Feature Selection

1: 1-classifier raw	92.1%	94.3%
<b>2: 1-classifier subtracted</b>	90.7%	<b>97.7%</b>
3: triple-classifier raw		
text	90.1%	92.6%
content	86.7%	80.0%
structure	73.0%	62.2%
combined	92.1%	90.9%
4: triple-classifier subtracted		
text	88.4%	90.7%
content	92.1%	92.1%
structure	80.0%	88.6%
<b>combined</b>	<b>95.4%</b>	93.0%

# Winning Strategy & Its Dataset

- RF goes into production
  - with baseline-subtracted features
  - single classifier
  - using preprocessing method (C)
- Trumps desired benefits of DTs
  - High accuracy rate
  - High consistency
- Sample size: 43
  - 22 true, 21 false
  - Dataset not skewed in one direction
- Selected Features:
  - Total pixel area of images
  - Number of
    - Words in page text
    - Words in title
    - “logout” keyword variants
    - “password” keyword variants
    - Divs
    - Input fields
    - Images
    - Links
    - Distinct colors
    - Cookies
    - Secure cookies

# SYSTEM ARCHITECTURE & DESIGN DECISIONS

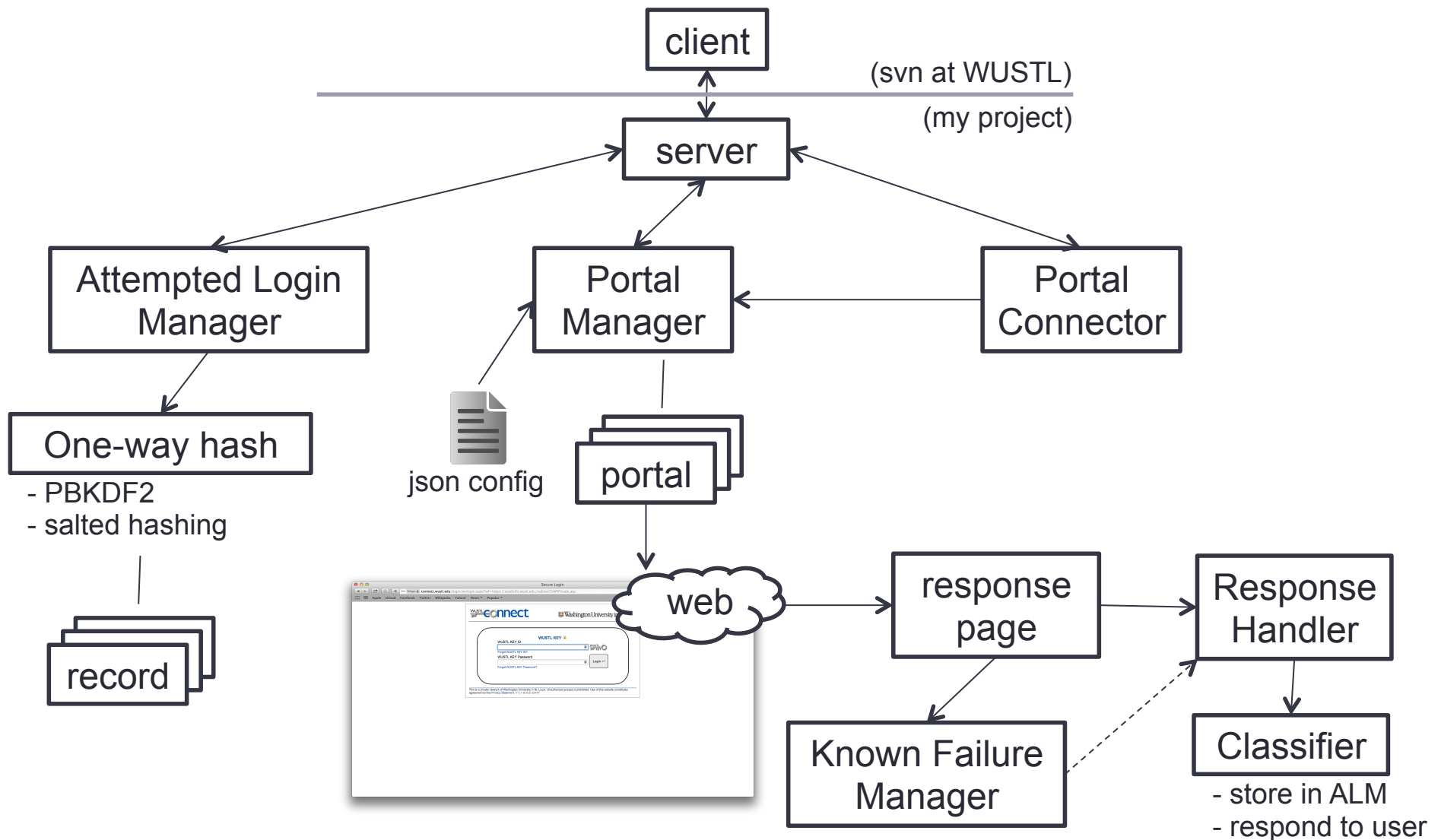
---



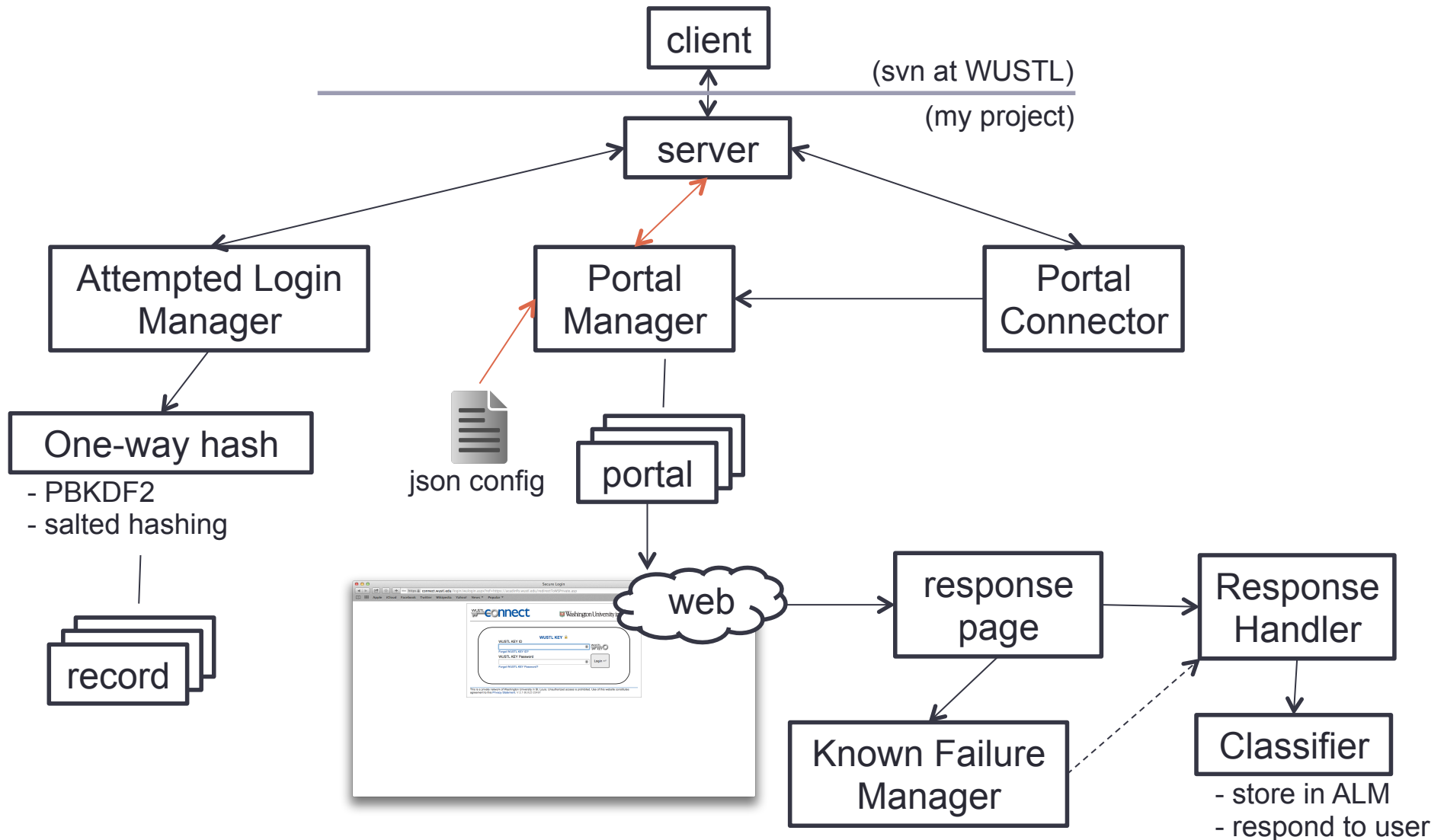
# Components

- HtmlUnit Headless Browser – web page interaction
  - Fullest feature-set of explored options
    - Javascript, Ajax support. Full webkit-backend.
  - Can't navigate everything on the web
    - Incompatible technologies: e.g. Flash, some non-standard JS libraries
    - Inability to parse ill-formed or relaxed syntax
- WEKA
- Complete list of 3<sup>rd</sup> party libraries listed in Appendix

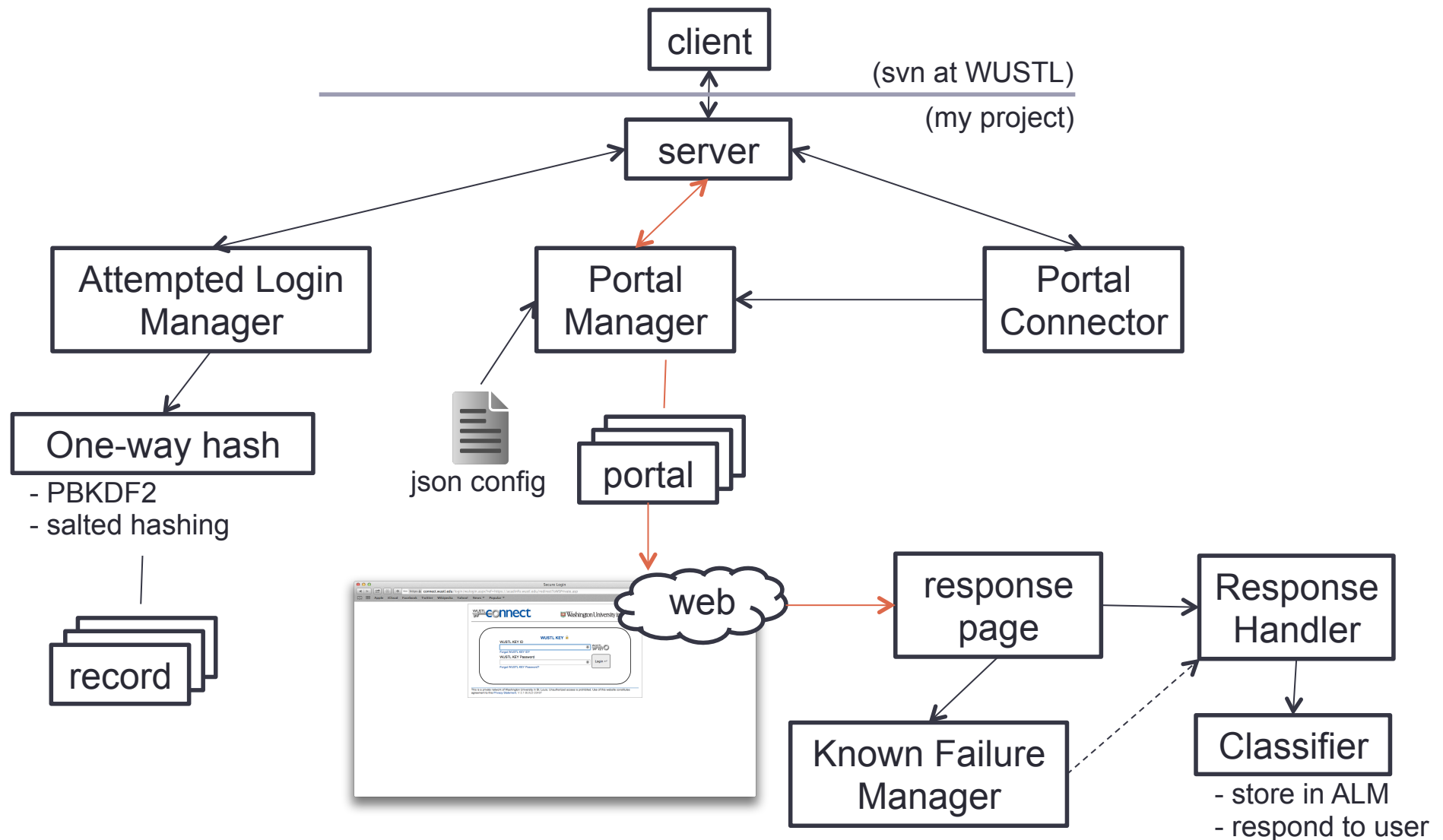
# Overview & Lifecycle of a Request



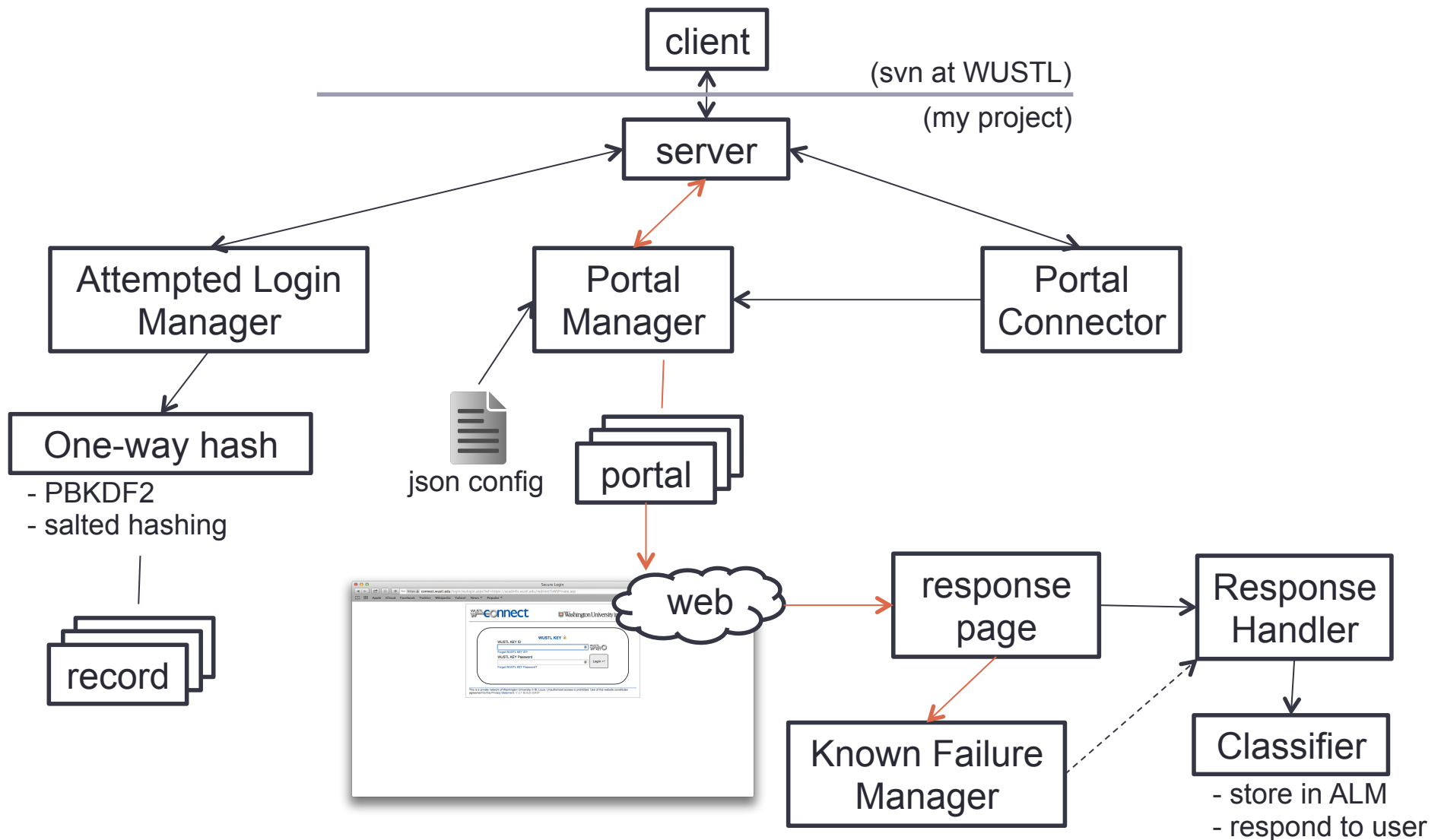
# Overview & Lifecycle of a Request



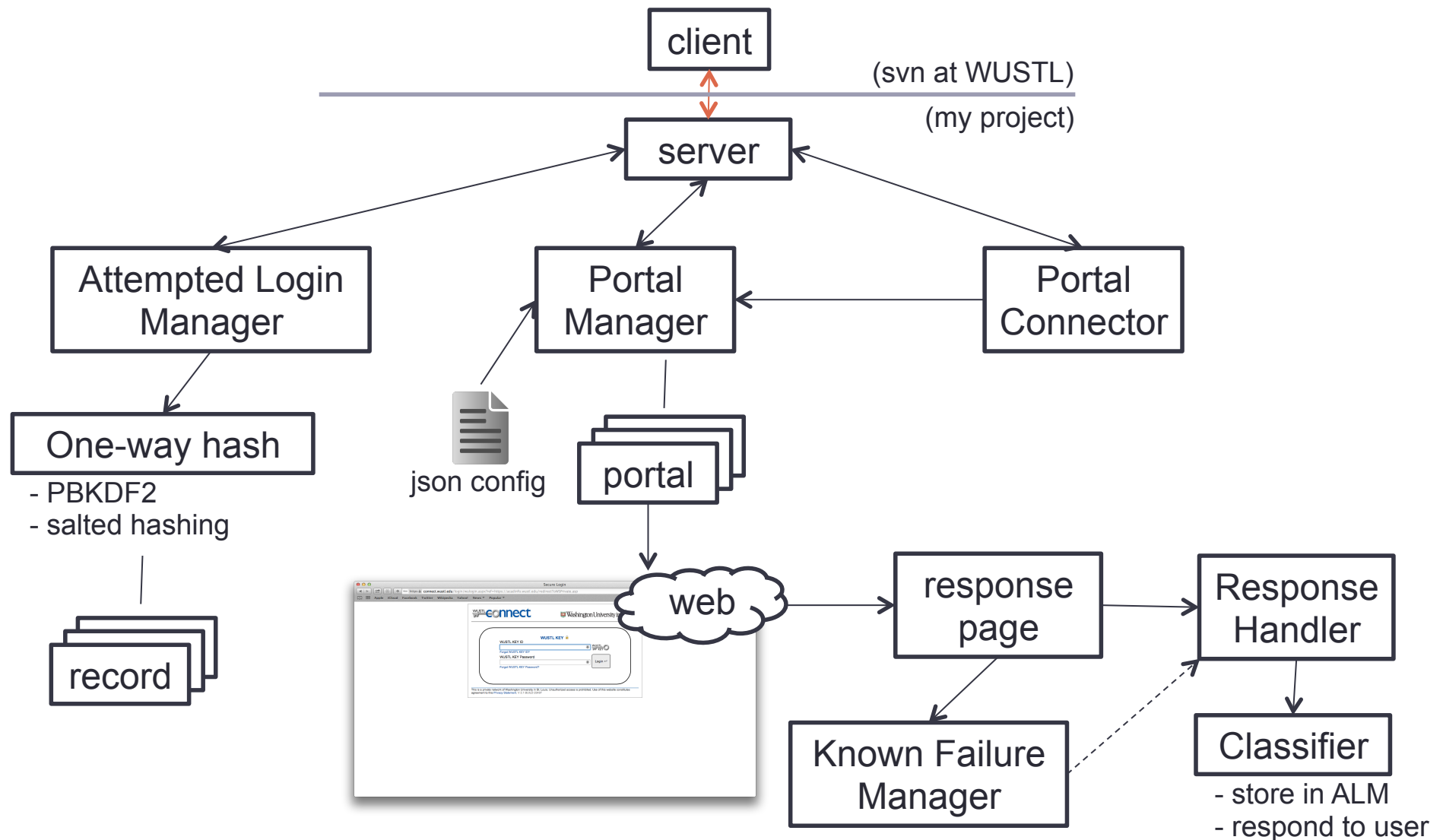
# Overview & Lifecycle of a Request



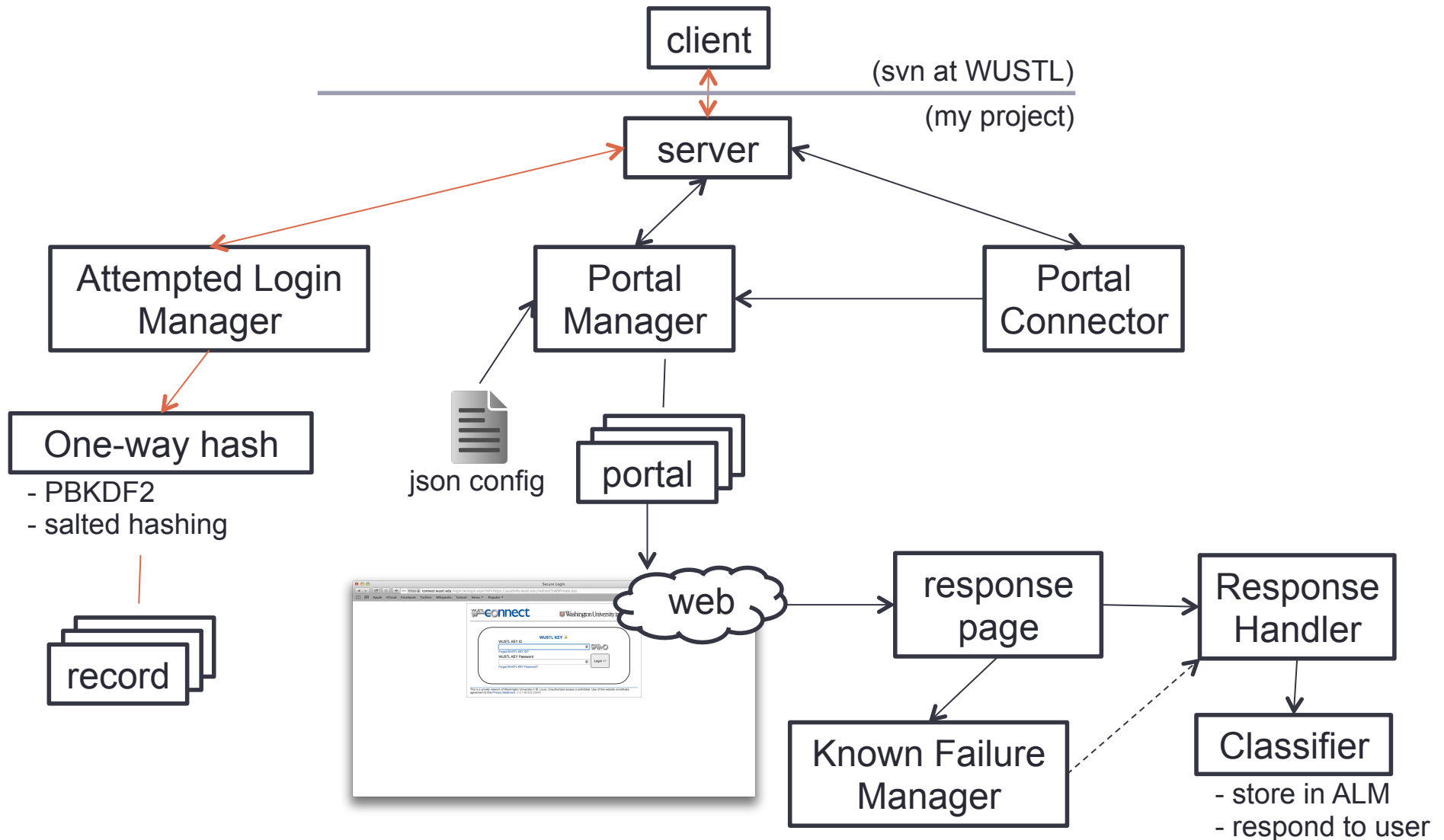
# Overview & Lifecycle of a Request



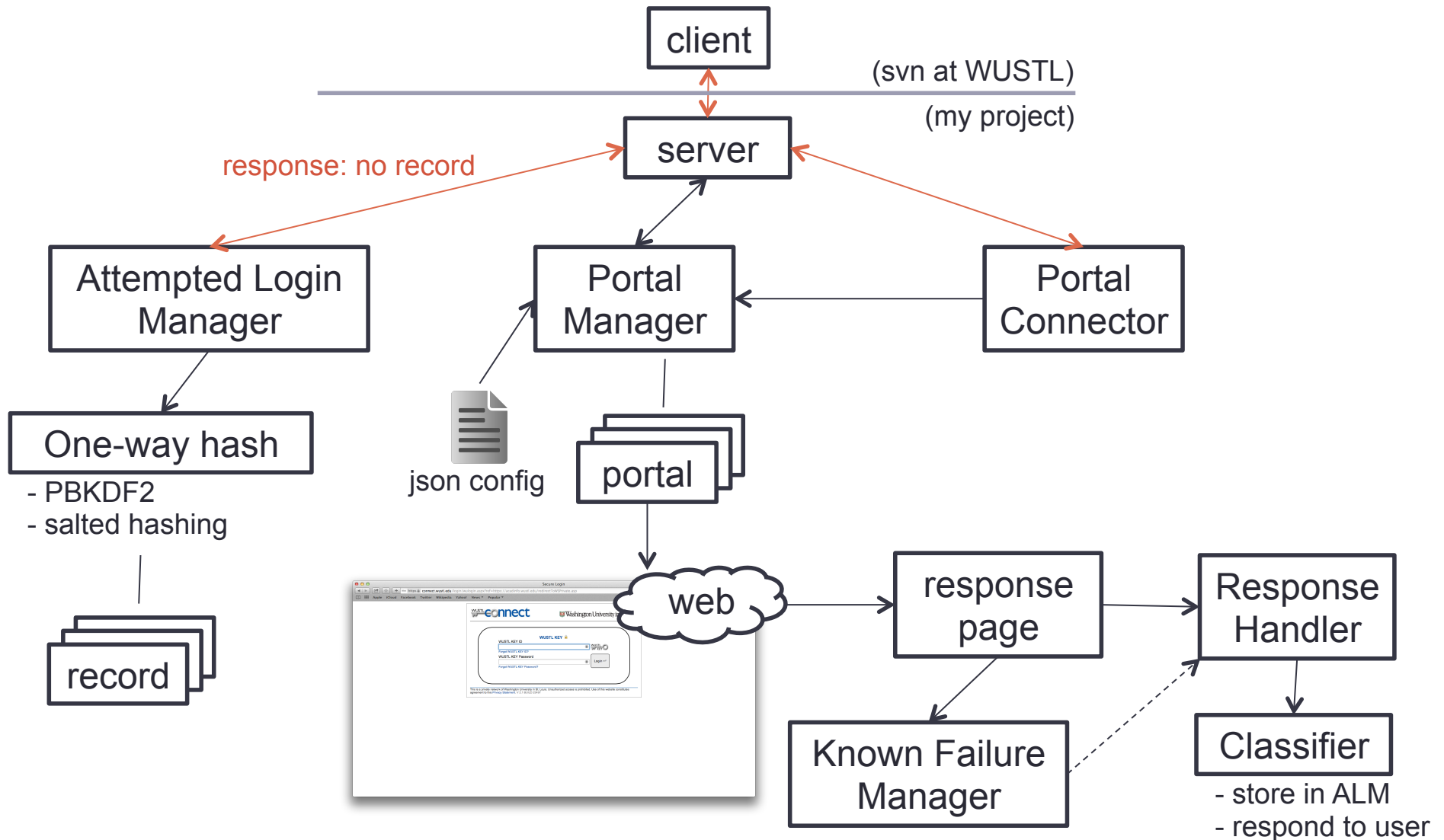
# Overview & Lifecycle of a Request



# Overview & Lifecycle of a Request

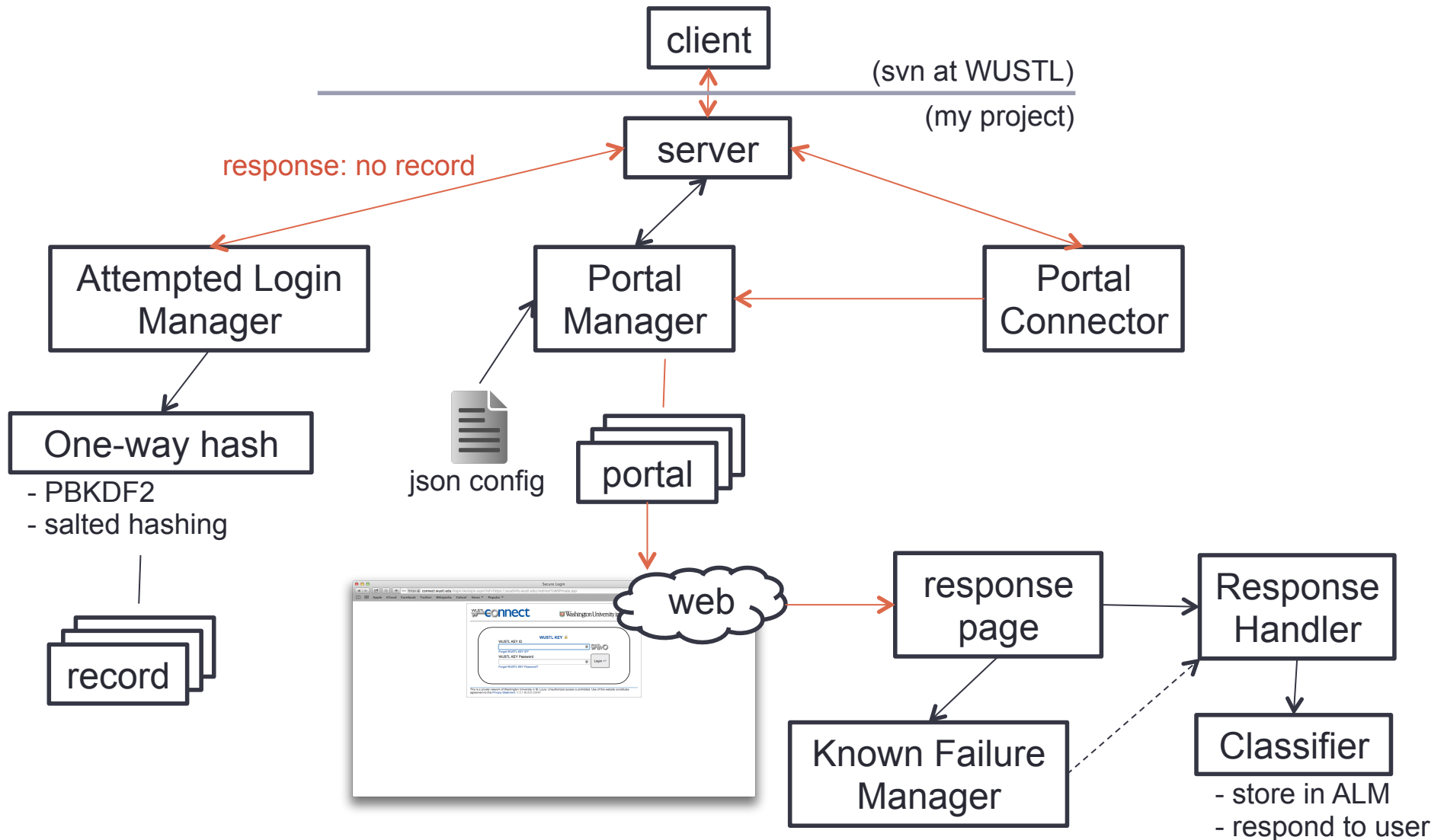


# Overview & Lifecycle of a Request

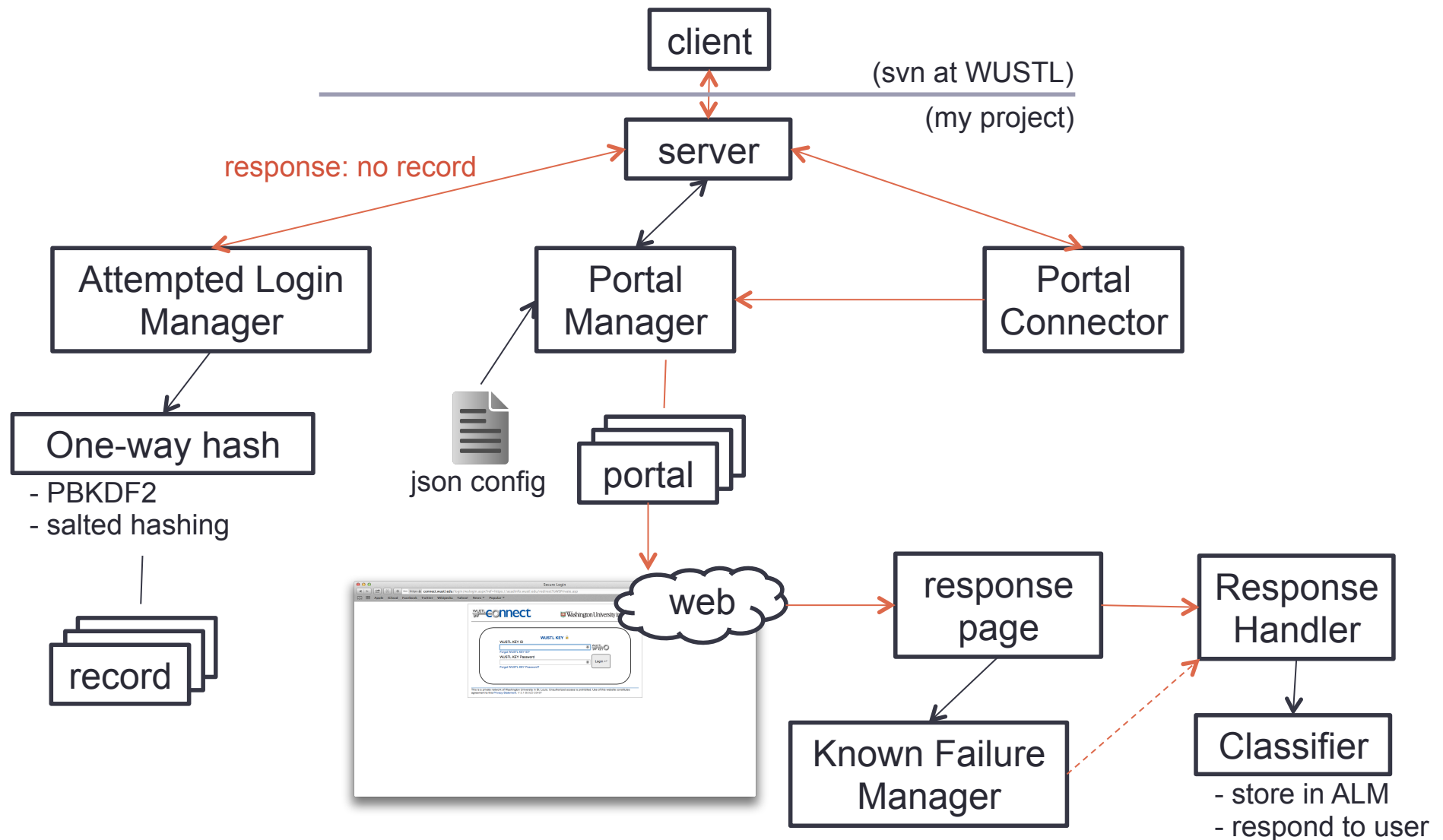




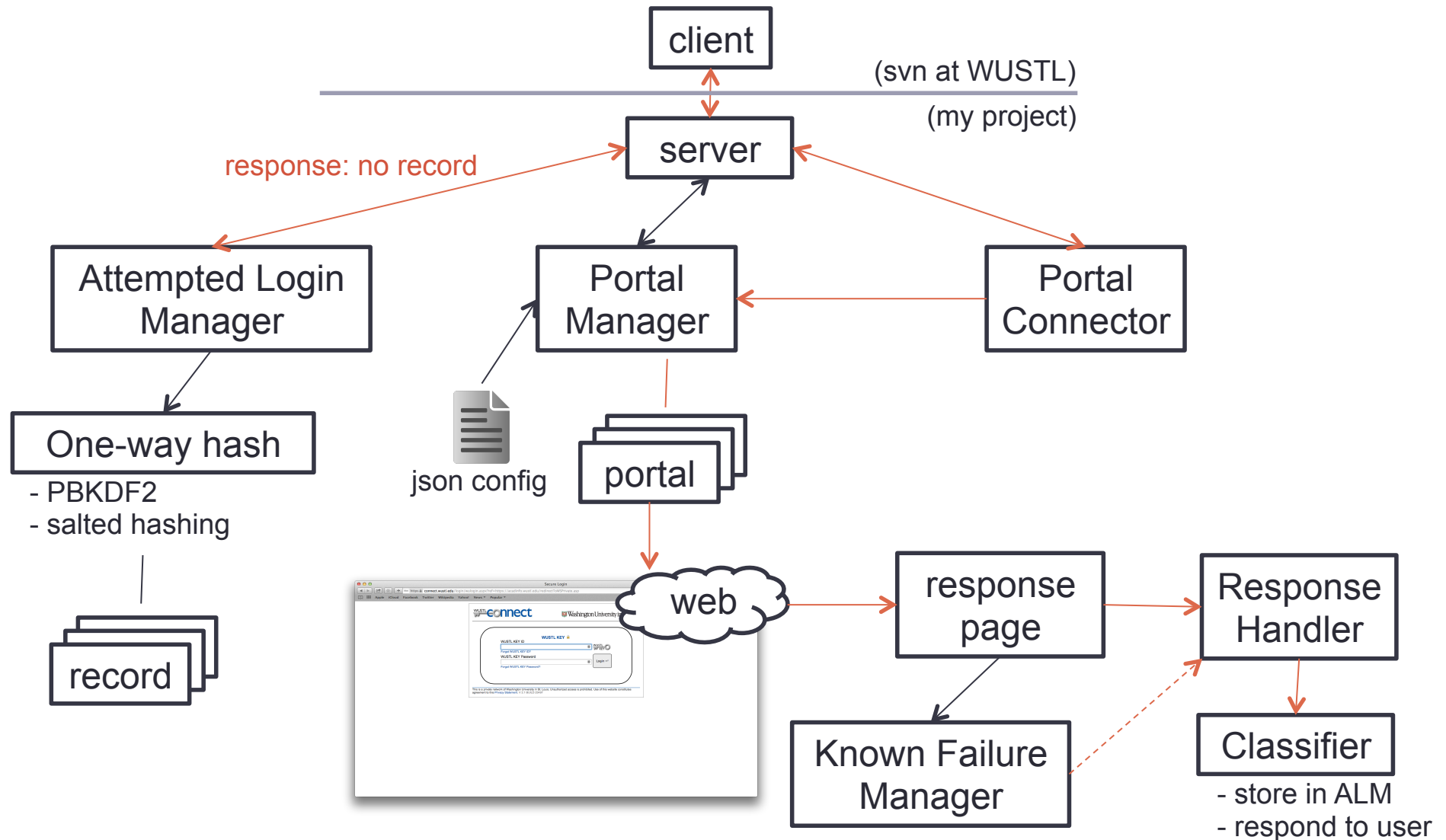
# Overview & Lifecycle of a Request



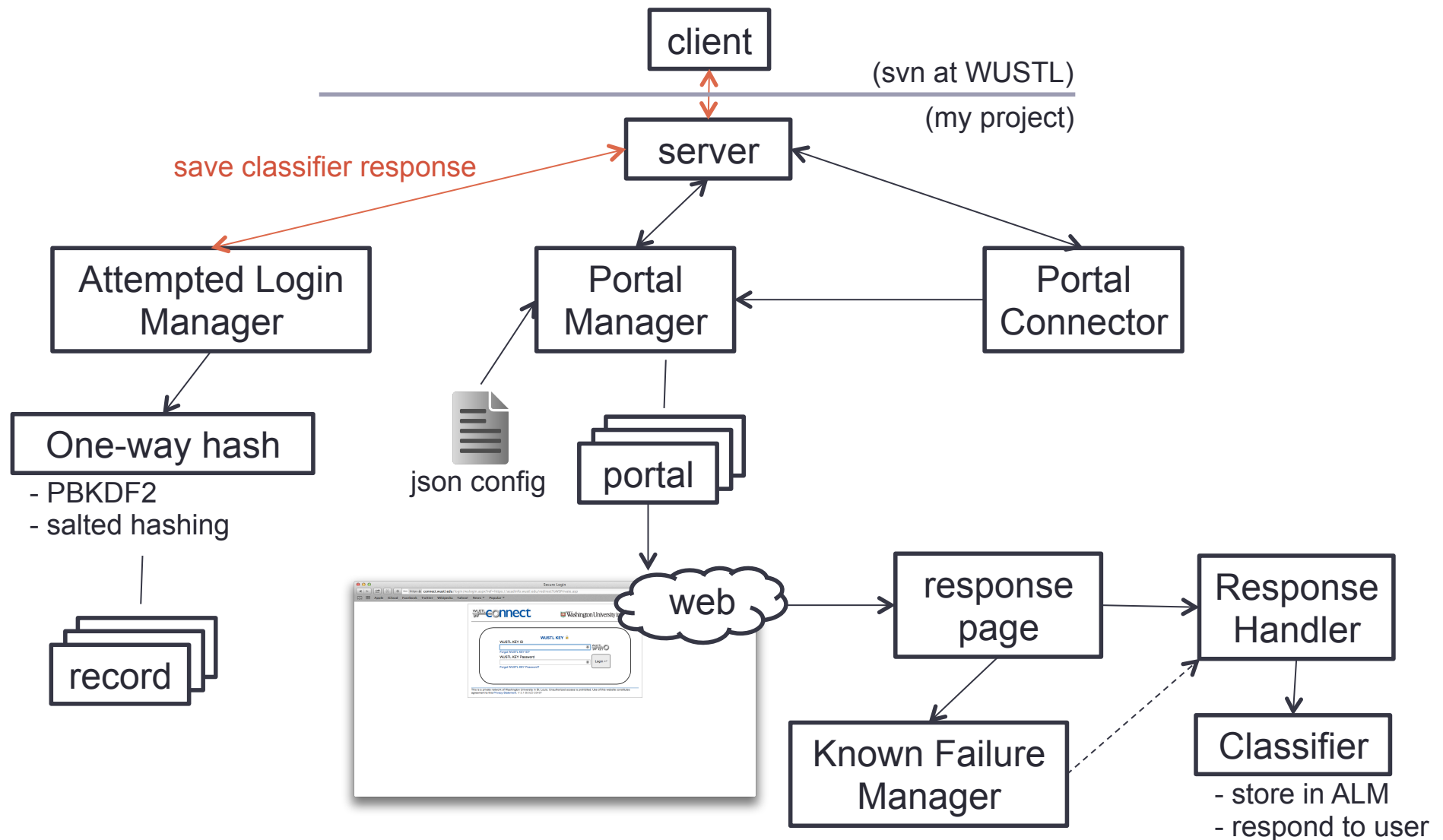
# Overview & Lifecycle of a Request



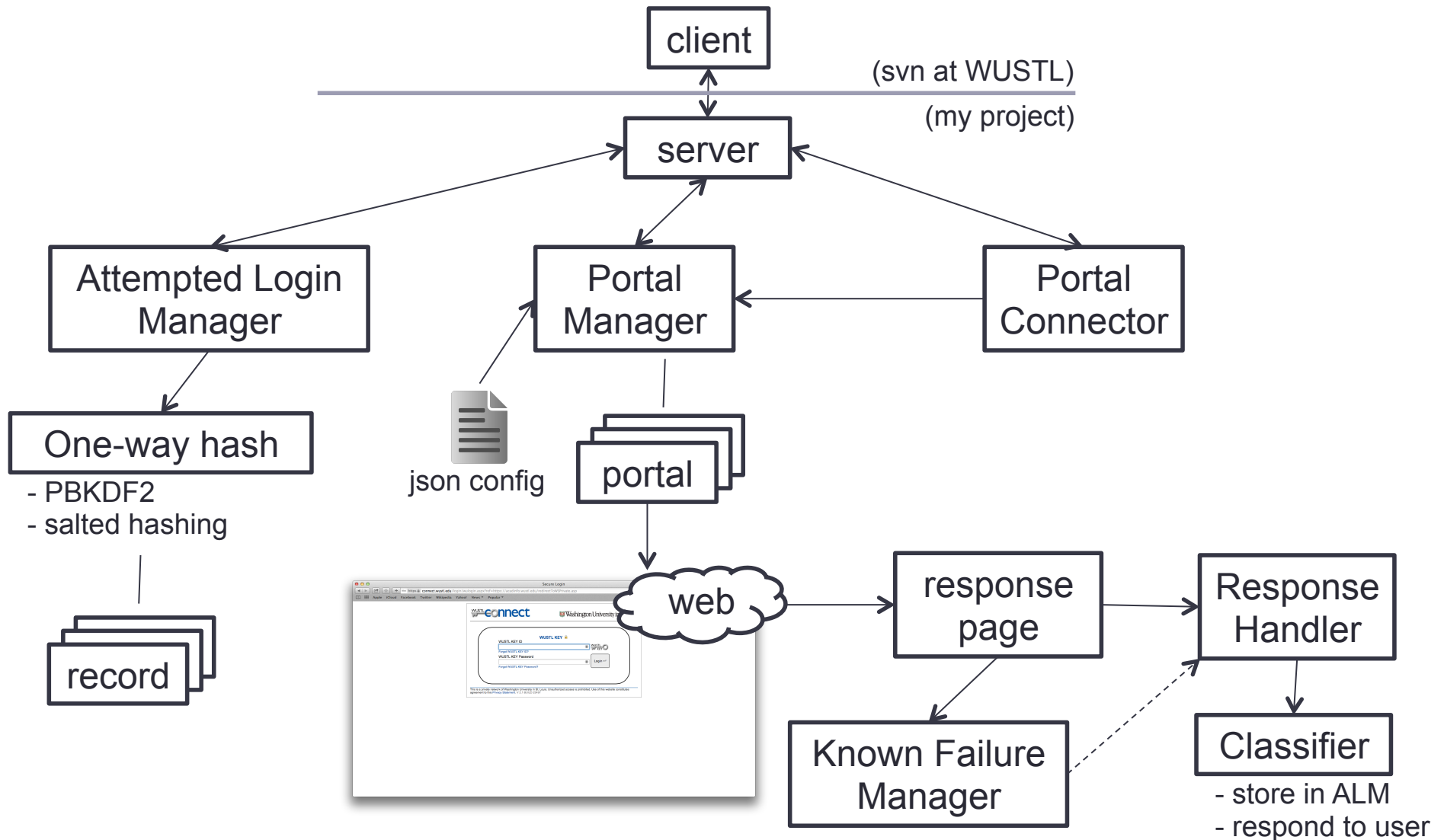
# Overview & Lifecycle of a Request



# Overview & Lifecycle of a Request



# Overview & Lifecycle of a Request



# Production Testing: Depth

- Server authentication accuracy on WUSTL's WebSTAC
  - Tested using 10 different accounts
  - 2 accounts agreed to generate & test lockout signals
  - 1 administrator account

USER	TRUTH	RESPONSE	TRUTH	RESPONSE	TRUTH	RESPONSE	SUCCESS?
Student_1	TRUE	TRUE	FALSE	FALSE	LOCKOUT (true pass)	FALSE	Yes
Student_2	TRUE	TRUE	FALSE	FALSE	LOCKOUT (true pass)	FALSE	Yes
Student_3	TRUE	TRUE	FALSE	FALSE			Yes
Student_4	TRUE	TRUE	FALSE	FALSE			Yes
Student_5	TRUE	TRUE	FALSE	FALSE			Yes
Student_6	TRUE	TRUE	FALSE	FALSE			Yes
Student_7	TRUE	TRUE	FALSE	FALSE			Yes
Student_8	TRUE	TRUE	FALSE	FALSE			Yes
Student_9	TRUE	TRUE	FALSE	FALSE			Yes
Admin_1	TRUE	TRUE	FALSE	FALSE			Yes

- General model works perfectly for WebSTAC

# Production Testing: Breadth

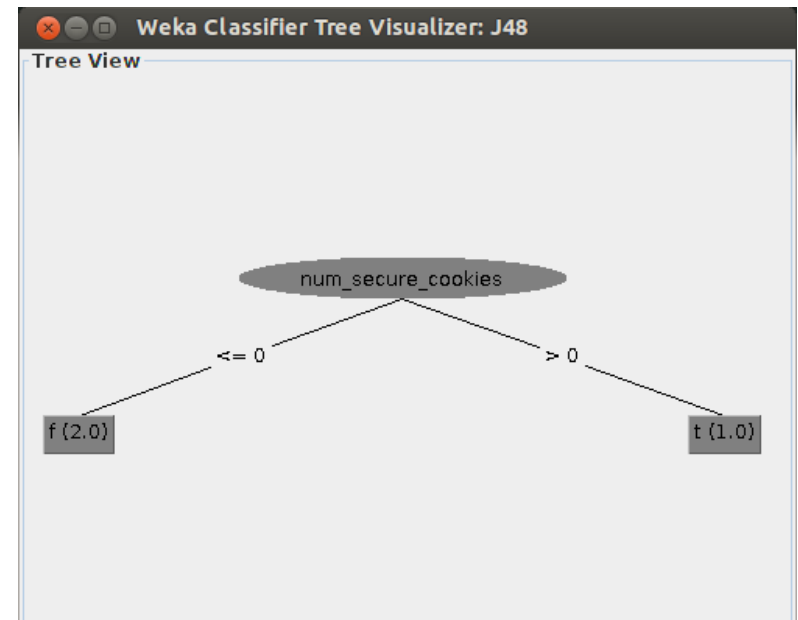
- Accuracy on 5 other websites
  - Created a single account at each site for testing

SITE	TRUTH	RESPONSE	TRUTH	RESPONSE	TRUTH	RESPONSE	SUCCESS?
Dropbox	TRUE	TRUE	FALSE	FALSE	LOCKOUT (true pass)	FALSE	Yes
ESPN	TRUE	TRUE	FALSE	FALSE	NO LOCKOUT	N/A	Yes
Piazza	TRUE	TRUE	FALSE	FALSE	LOCKOUT (true pass)	FALSE	Yes
Ebay	TRUE	TRUE	FALSE	FALSE	LOCKOUT (true pass)	FALSE	Yes
Twitter	TRUE	TRUE	FALSE	FALSE	LOCKOUT (true pass)	TRUE	No

- Model worked perfectly on 4 of 5 sites
  - Twitter worked well except for its lockout signal
- In general, reasonable success
  - The case for Twitter...

# The Case for Twitter

- What if an admin really does know everything about a site?
  - Example: Goal is to authenticate very well at Twitter *only*
- Allow the admin to specify
  - Admin collects features using scraping tool
  - Admin uses json config to use site-specific model
  - A simple DT model is created for site
- Not a good machine learning problem
  - Nevertheless, still useful in practice
  - End-user flexibility





# DEMO

---

<https://www.auto-auth-server.com/>

# AREAS FOR IMPROVEMENT & LESSONS LEARNED

---

# Areas for Improvement

- Technical Shortfalls of HtmlUnit
  - Can't handle some technologies
  - Mobile site versions were fairly helpful in circumventing this
- Feature Refinement
  - More advanced keyword tagging, sentiment analysis
  - Further feature research
- Data Mining Experiments & Data Gathering
  - Testing other types of classifiers
  - Continuous learning
- General Project Improvement
  - Expansion & end-user flexibility
    - Captcha handling, Two-step login processes
  - Environmental enhancement: resources in the cloud

# Lessons Learned

- Reflection & Databases
  - Bad idea & the source of many headaches
  - Should have used web project CRUD design patterns
- Thoroughness of data collection processes
  - Recollected / retested LOOCV several times
- Bad auth practices on the web & surprises
  - Great security (Reddit, Steam ...)
    - Captchas, IP blocking, increasing attempt duration.
  - Less than ideal (ESPN, Paypal, eHarmony...)
    - Some sites simply never lock an account (~30 attempts per site)
    - Paypal seemed to assist its lockout mechanism using cookies

# THANK YOU

---

Open to Questions

## References & Resources






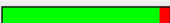

- Chau, Michael & al. 2007. *A machine learning approach to web page filtering using content and structure analysis.*
- Eshete, Birhanu. 2013. *Effective Analysis, Characterization, and Detection of Malicious Web Pages.*
- Herzog, Christoph & al. 2013. *Feature-based Object Identification for Web Automation.*
- Indra Devi, M. & al. 2007. *Machine Learning Techniques for Automated Web Page Classification using URL Features.*
- Morgan, Timothy D. 2010. *Weaning the Web off of Session Cookies: Making Digest Authentication Viable.*
- Sug, Hyontai. 2013. *Insights of Data Mining for Small and Unbalanced Data Set Using Random Forests.*

# Appendix

- 3<sup>rd</sup> Party Libraries
  - Apache Commons IO 2.4
  - AlchemyAPI 0.8
  - Cloning 1.8.5
  - HtmlUnit 2.11
  - JavaMail 1.5.1
  - Jersey 1.18
  - Joda-Time 2.3
  - Json-smart 2.0
  - MySQL Connector/J 5.1.29
  - Weka 3.7.10

# Appendix

- JUnit Tests
  - All tests passed
  - Code Coverage: over 90%

Show: Application classes ▾		<a href="#">Settings</a>			Metrics for: auto-auth-server	
Elem	Cov%	Av Me Cpx	▼	Complexity	Structure	Test Executions
▼ auto-auth-server	 91.6%	2.2		312.0	Packages: 6	Executed Tests: 0
▶ main	 92.7%	2.2		13.0	Files: 23	Passes: 0
▶ util	 70.0%	3.6		25.0	Classes: 26	Fails: 0
▶ server_mgmt	 100.0%	1.1		34.0	Methods: 145	Errors: 0
▶ persistence	 97.5%	1.7		48.0	Statements: 700	
▶ portals	 94.1%	1.8		88.0	Branches: 218	
▶ learning	 90.4%	4.2		104.0		
					Source	
					LOC: 1,978	NC LOC: 1,494
					Total Cmp: 312	Cmp Density: 0.4
					Avg Method Cmp: 2.2	