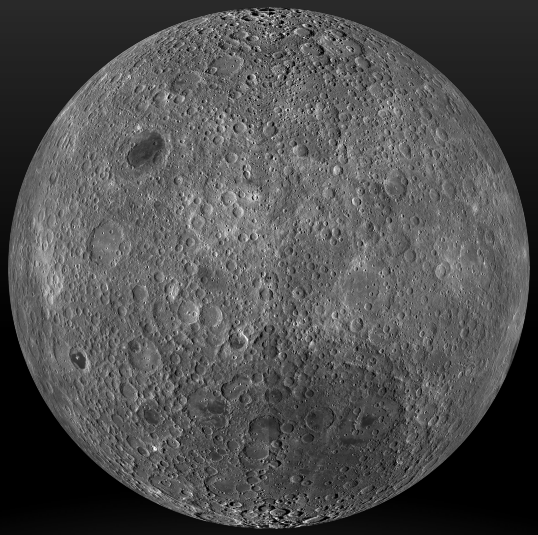


Selenium

dark side of the moon

Adam Reiser

https://github.com/adamreiser/selenium

Outline

- Who am I?
- What is Selenium?
- Context and ecosystem
- Using Selenium offensively
- Pitfalls and when not to use Selenium

Who am I?

- Adam Reiser
- Security Researcher at Cisco
- Penetration testing
- Product security assessments
  - custom web app attacks

What is Selenium?

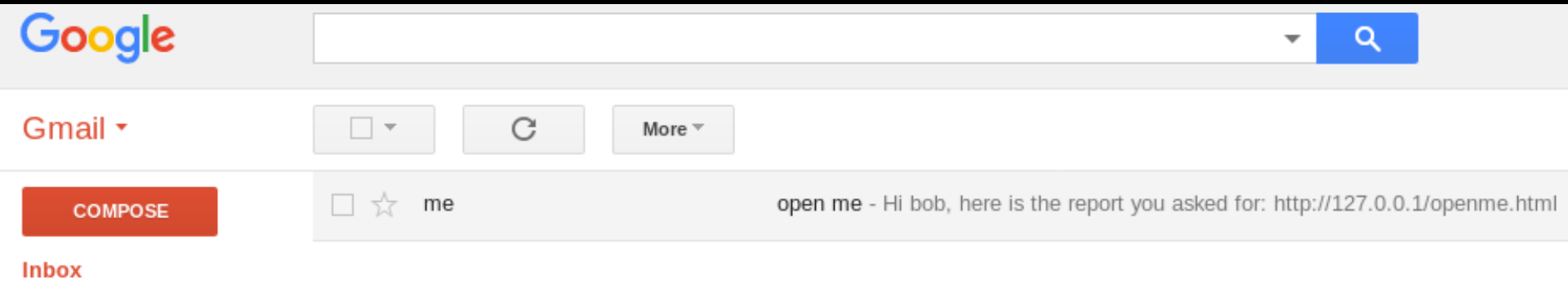
- Set of tools to programatically control a browser
- Widely used in web application testing
- Automates user actions such as keystrokes and clicks
- Compatible across many browsers and languages

Selenium ecosystem

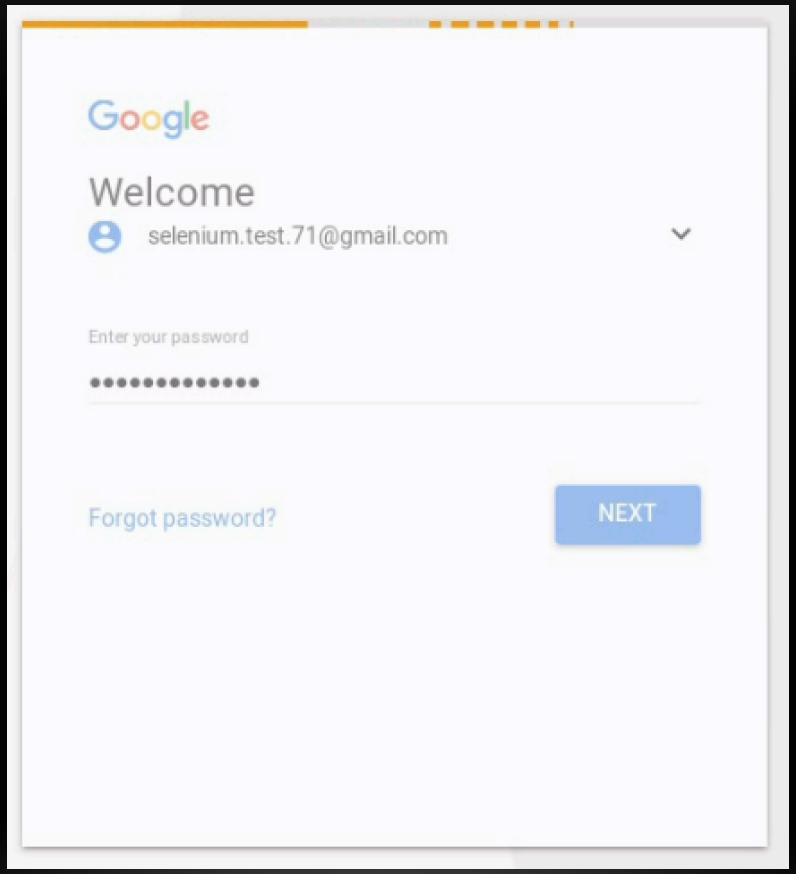
- Many browser automation projects
- Usually based on (or support) WebDriver
- Native headless support in major browsers
- Active development and community

Demo

- Testing a client-side attack
- Open an email and click on a link



Demo

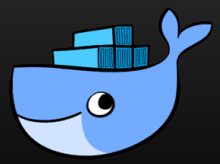


gmail.py

Components and toolkit

- Browser to automate (stock/unmodified)
  - Corresponding WebDriver
- Selenium libraries
- A DOM inspector
- Application proxy
- Lab environment

Lab setup



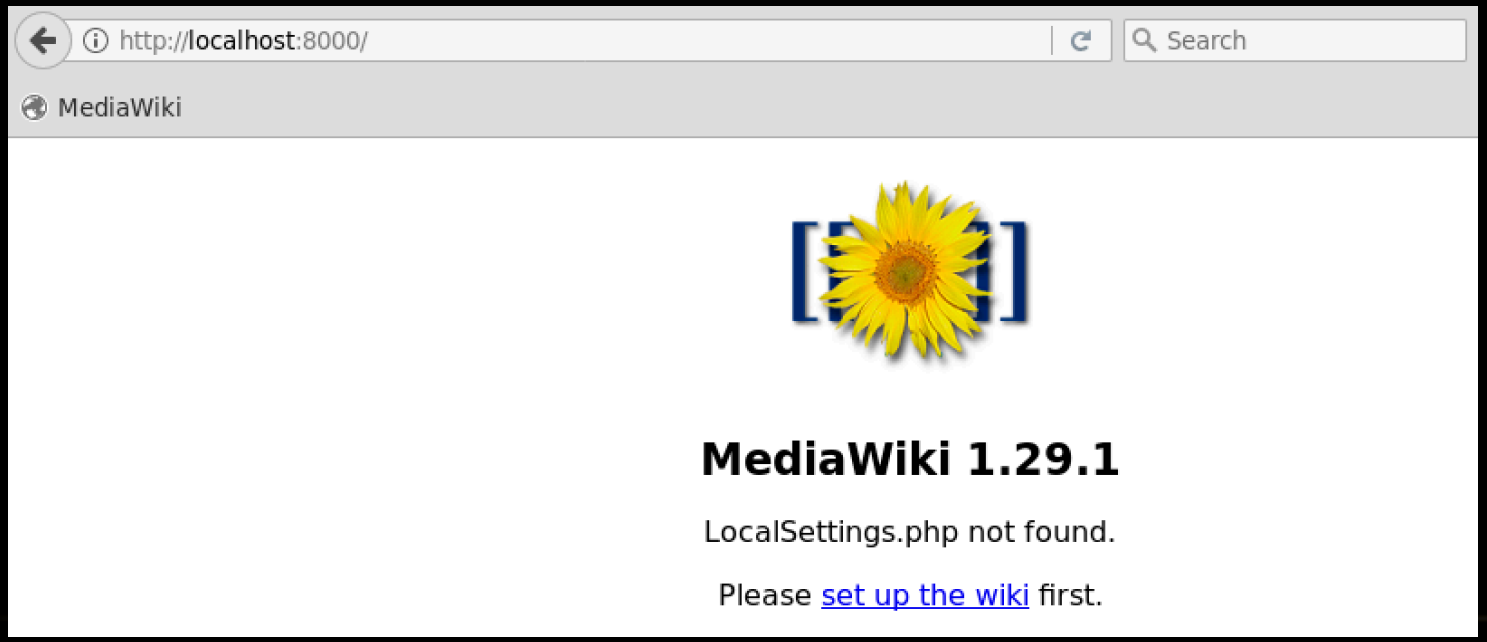
- Docker setup for x64 kali (testing)
  - https://docs.docker.com/engine/installation/linux/docker-ce/debian/
- docker-compose binary for your PATH
  - https://docs.docker.com/compose/install/#install-compose

Lab setup

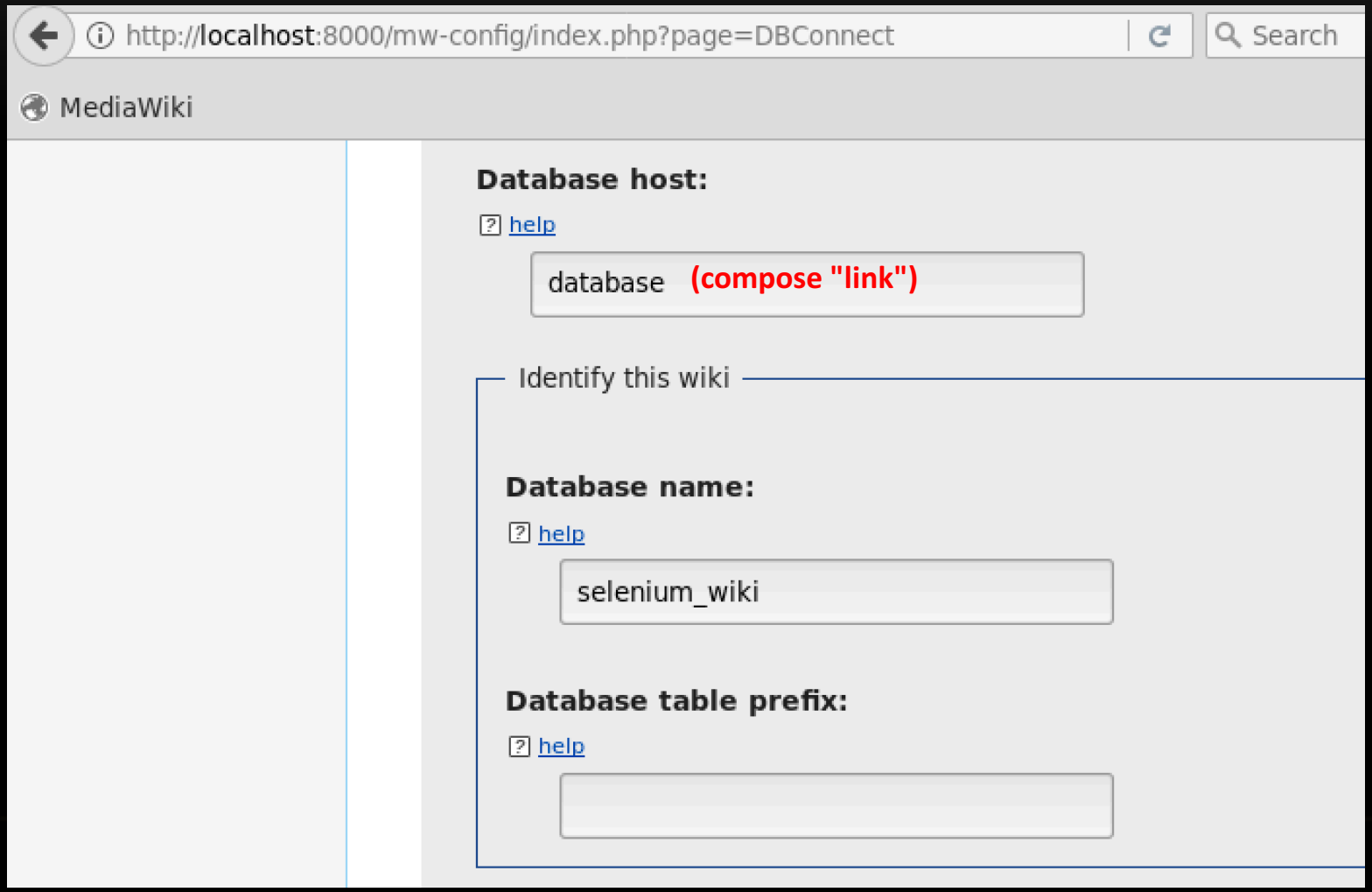
root@kali:~# git clone https://github.com/adamreiser/selenium.git

root@kali:~# cd selenium/mediawiki

root@kali:~/selenium/mediawiki# docker-compose up



Lab setup

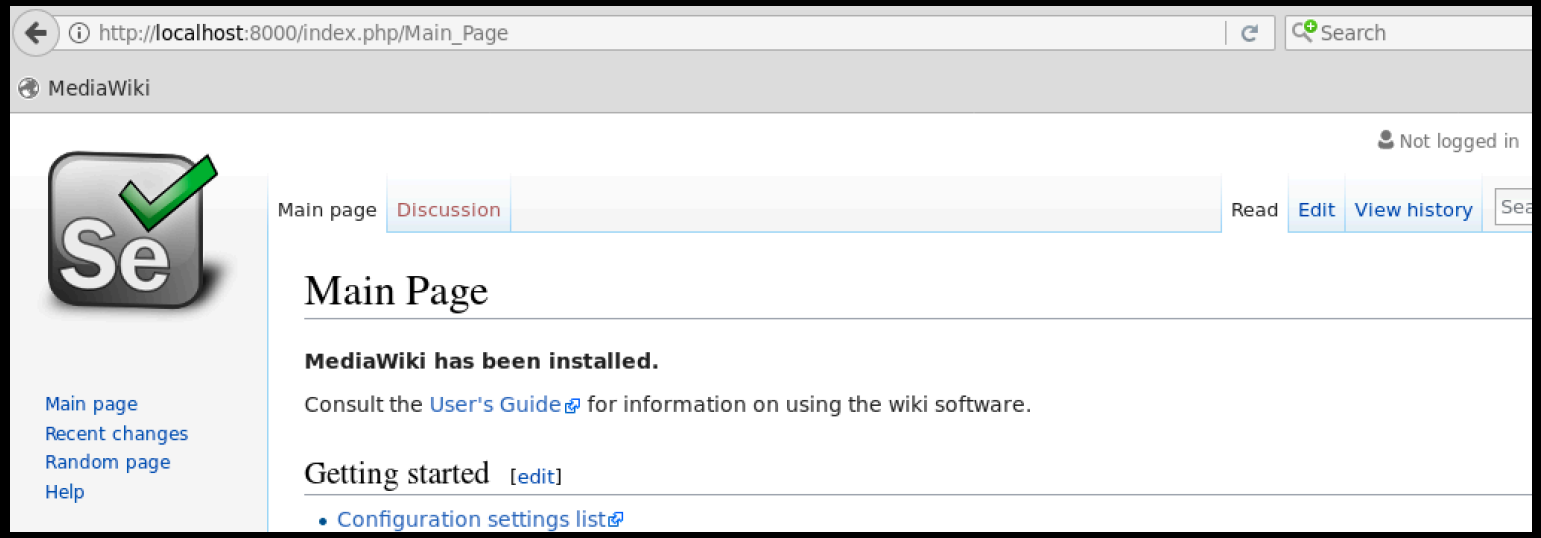

http://localhost:8000/mw-config/index.php?page=DBConnect
Search
MediaWiki
Database host:
help
database (compose "link")
Identify this wiki
Database name:
help
selenium_wiki
Database table prefix:
help

Lab setup

- Download resulting LocalSettings.php
  - Copy to selenium/mediawiki
- Ctrl+C (gracefully) terminates compose
- mediawiki/database will be populated
- Uncomment from docker-compose.yml
  - #- ./LocalSettings.php:/var/www/html/LocalSettings.php

Lab setup

root@kali:~/selenium/mediawiki# docker-compose up



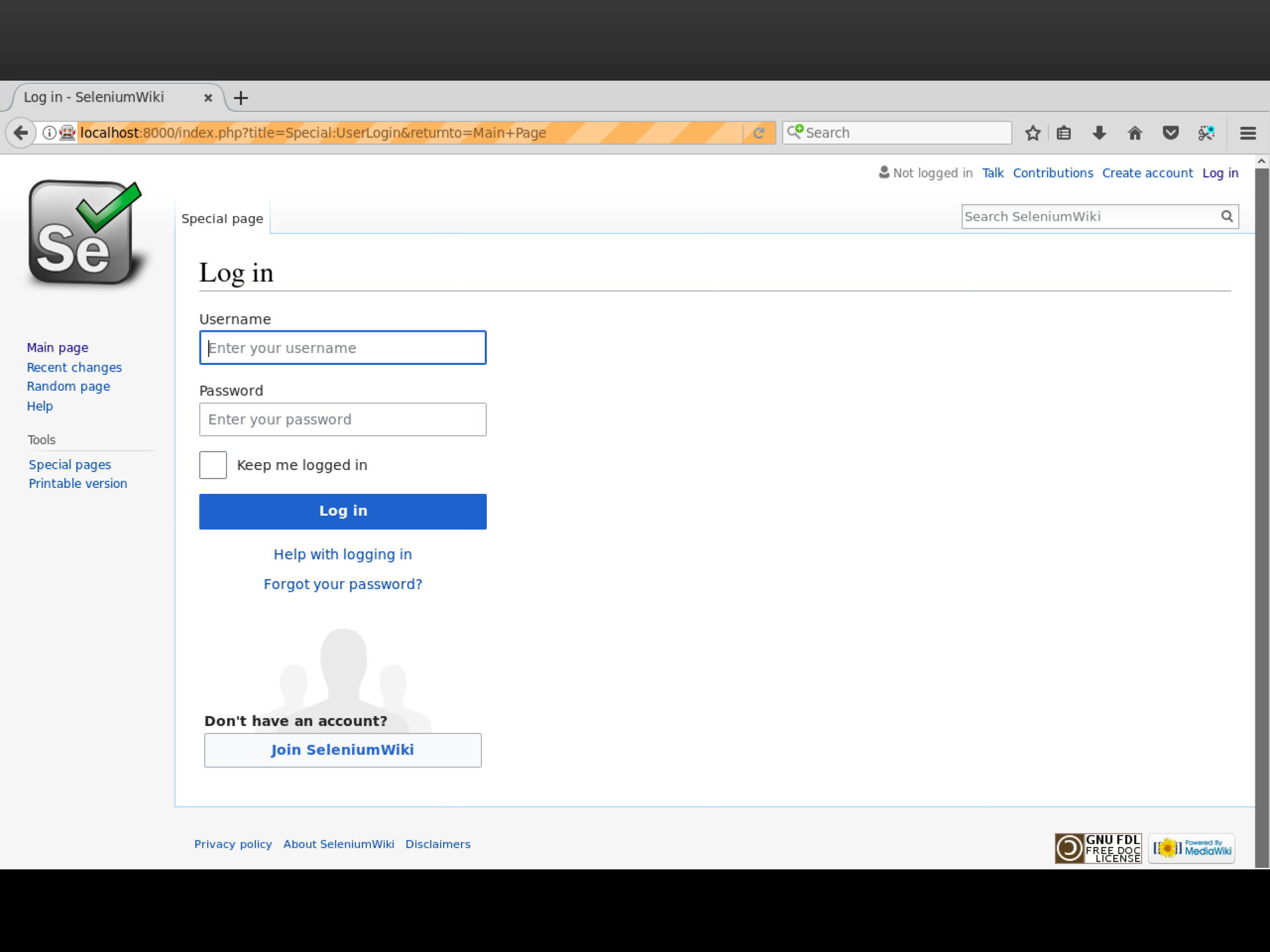
Sketching a Selenium script

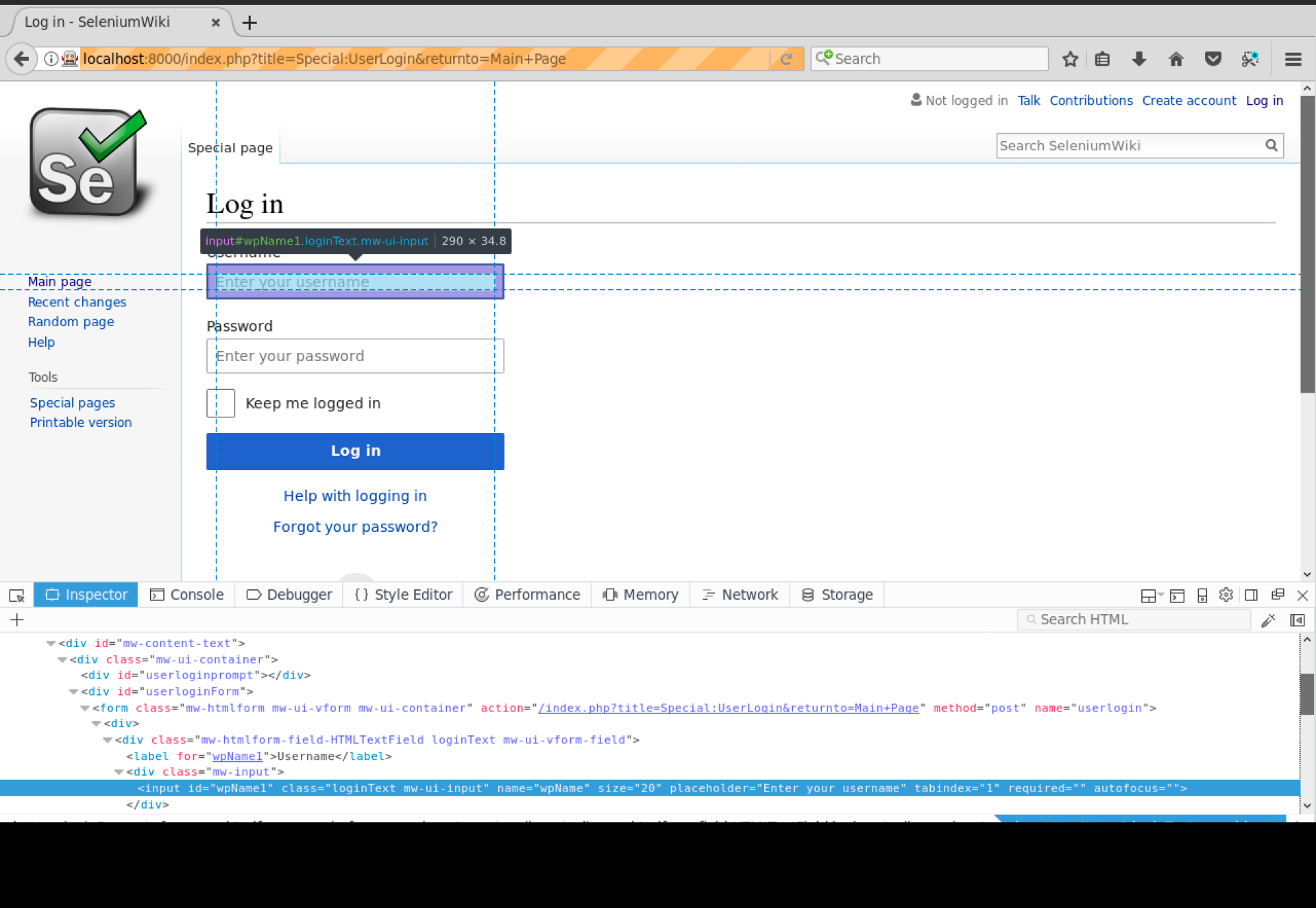
```
#!/usr/bin/env python
from selenium import webdriver
import atexit
import tabcomplete

wd = webdriver.Firefox()
atexit.register(wd.quit)
wd.implicitly_wait(30)
wd.get("http://localhost:8000")
```

onesmallstep.py

Sketching a Selenium script

```
root@kali:~/selenium/# python -i onesmallstep.py
>>> wd.find_element_by_link_text("Log in").click()
```


Log in - SeleniumWiki
localhost:8000/index.php?title=Special:UserLogin&returnto=Main+Page
Search
Not logged in Talk Contributions Create account Log in
Special page
Search SeleniumWiki
Log in
Username
Enter your username
Password
Enter your password
Keep me logged in
Log in
Help with logging in
Forgot your password?
Main page
Recent changes
Random page
Help
Tools
Special pages
Printable version
Don't have an account?
Join SeleniumWiki
Privacy policy
About SeleniumWiki
Disclaimers
GNU FDL FREE DOC LICENSE
Powered By MediaWiki


Log in - SeleniumWiki
localhost:8000/index.php?title=Special:UserLogin&returnto=Main+Page
Search
Not logged in Talk Contributions Create account Log in
Special page
Search SeleniumWiki
Log in
input#wpName1.loginText.mw-ui-input 290 × 34.8
Enter your username
Password
Enter your password
Keep me logged in
Log in
Help with logging in
Forgot your password?
Main page
Recent changes
Random page
Help
Tools
Special pages
Printable version
Inspector
Console
Debugger
Style Editor
Performance
Memory
Network
Storage
Search HTML

<form class="mw-htmlform mw-ui-vform mw-ui-container" action="/index.php?title=Special:UserLogin&returnto=Main+Page" method="post" name="userlogin">

<label for="wpName1">Username</label>

<input id="wpName1" class="loginText mw-ui-input" name="wpName" size="20" placeholder="Enter your username" tabindex="1" required="" autofocus="">

Sketching a Selenium script

```
root@kali:~/selenium/# python -i onesmallstep.py

>>> wd.find_element_by_link_text("Log in").click()

>>> wd.find_element_by_id("wpName1").send_keys(username)

>>> wd.find_element_by_id("wpPassword1").send_keys(password)

>>> wd.find_element_by_id("wpLoginAttempt").click()
```

Adding a proxy

- Suppose we encounter something like this:

```
<body onload="javascript:document.forms[0].submit()">
```

- Can we "pause" Selenium here?
- No framework for hooking browser events
- Proxy to the rescue

Adding a proxy

- Logging
- Perform on-the-fly request/response transforms
- Enforce scope
- Issue detection
- Halt a transaction

Adding a proxy

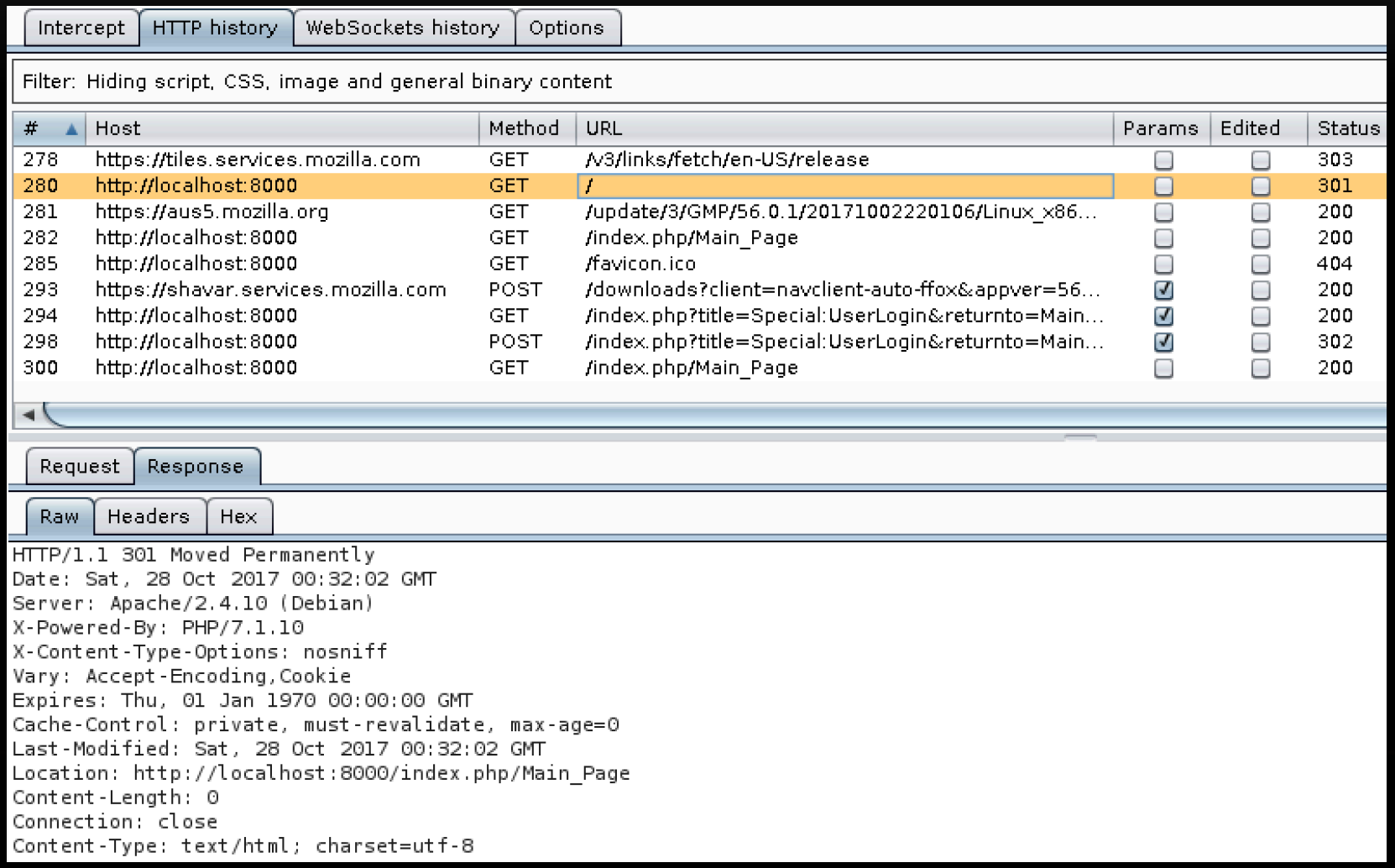
- Each Selenium run creates a fresh profile
- Preconfigure the new profile to use a proxy
- Install the proxy's certificate authority

Proxy (and preference) mechanics

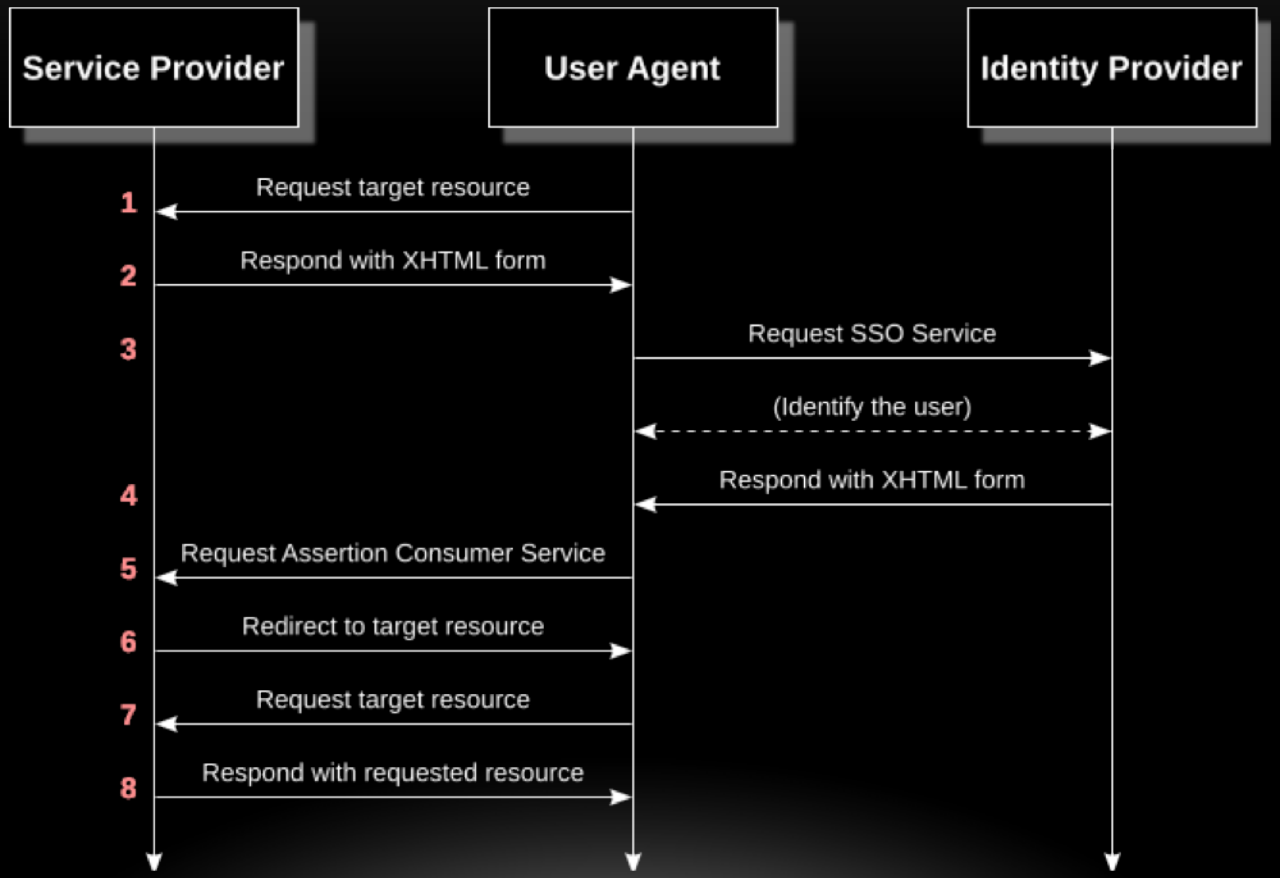
```
class SeleniumProxy():
    """Base class to set up Selenium with a proxy."""

    def __init__(self, ca_file)):

        fp = webdriver.FirefoxProfile()
        fp.set_preference("network.proxy.http", "127.0.0.1")
        fp.set_preference("network.proxy.http_port", 8080)
        fp.set_preference("network.proxy.ssl", "127.0.0.1")
        fp.set_preference("network.proxy.ssl_port", 8080)
        fp.set_preference("network.proxy.type", 1)
        fp.set_preference("network.proxy.no_proxies_on", "")

        shutil.copy(kwargs["ca_file"], fp.path)
        kwargs["firefox_profile"] = fp
        kwargs.pop("ca_file")
        self.wd = webdriver.Firefox(**kwargs)
```

seleniumprofile.py

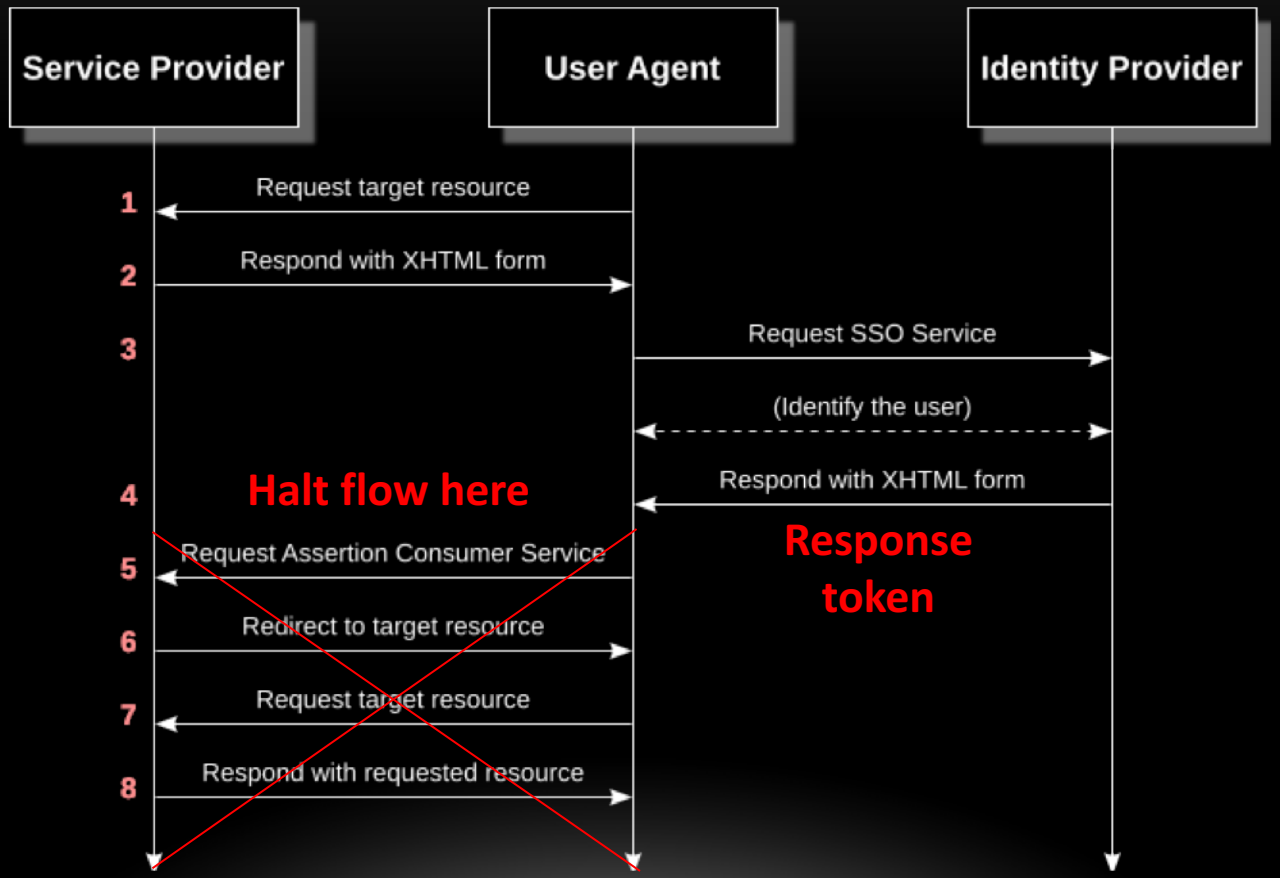
Proxy history



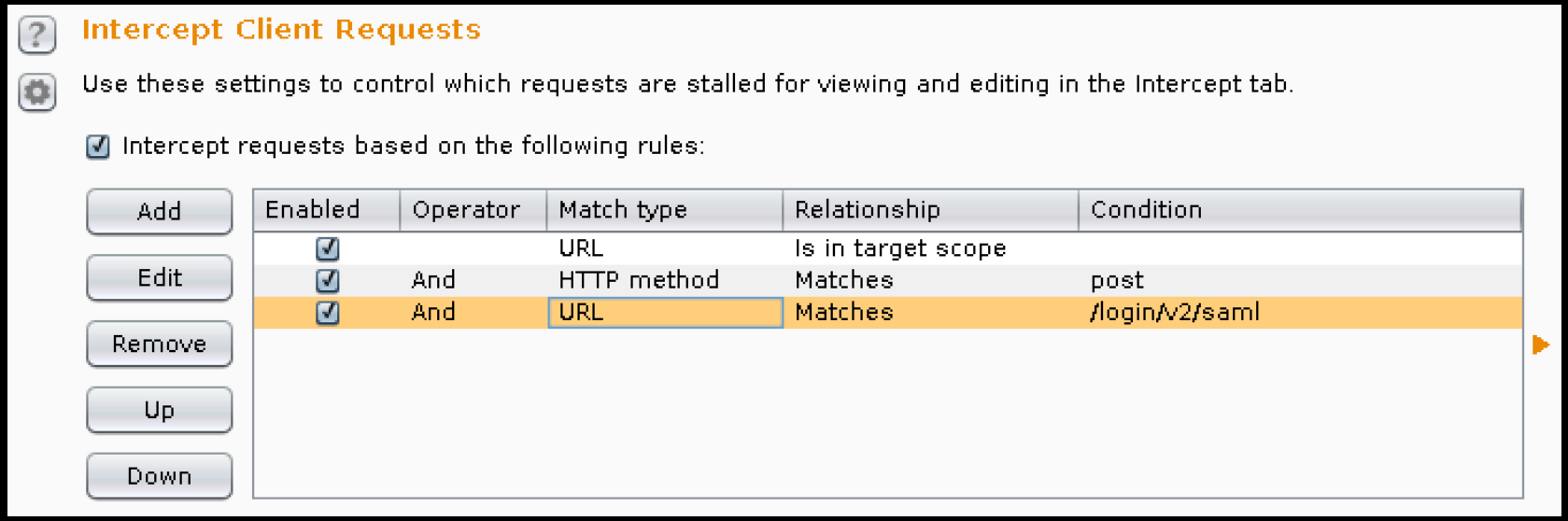
Token capture



Token capture



Token capture



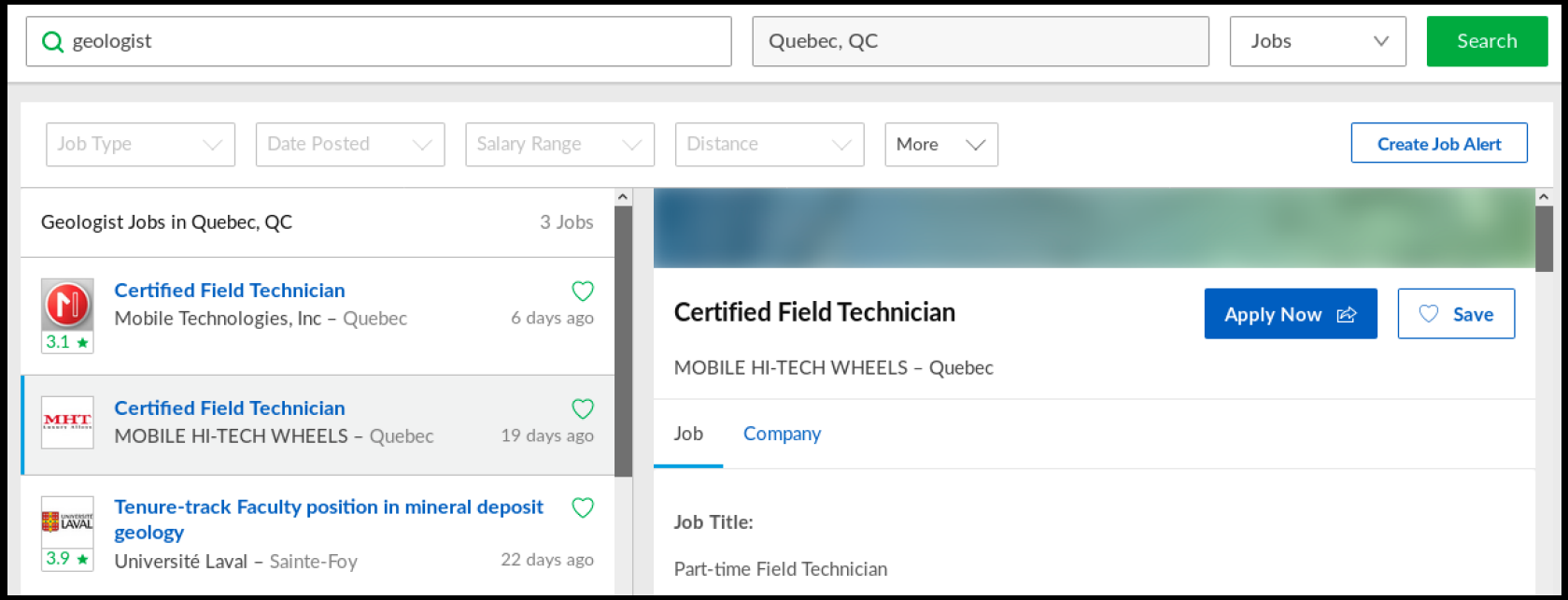
Gotchas

- Timing
  - Defining what you want
- XHR/Ajax
  - Elements "fading in"
- Composite actions
  - mouseover and click
- Bot detectors
  - aka *"people trying to stop you from doing exactly what you're trying to do"*
- Frames

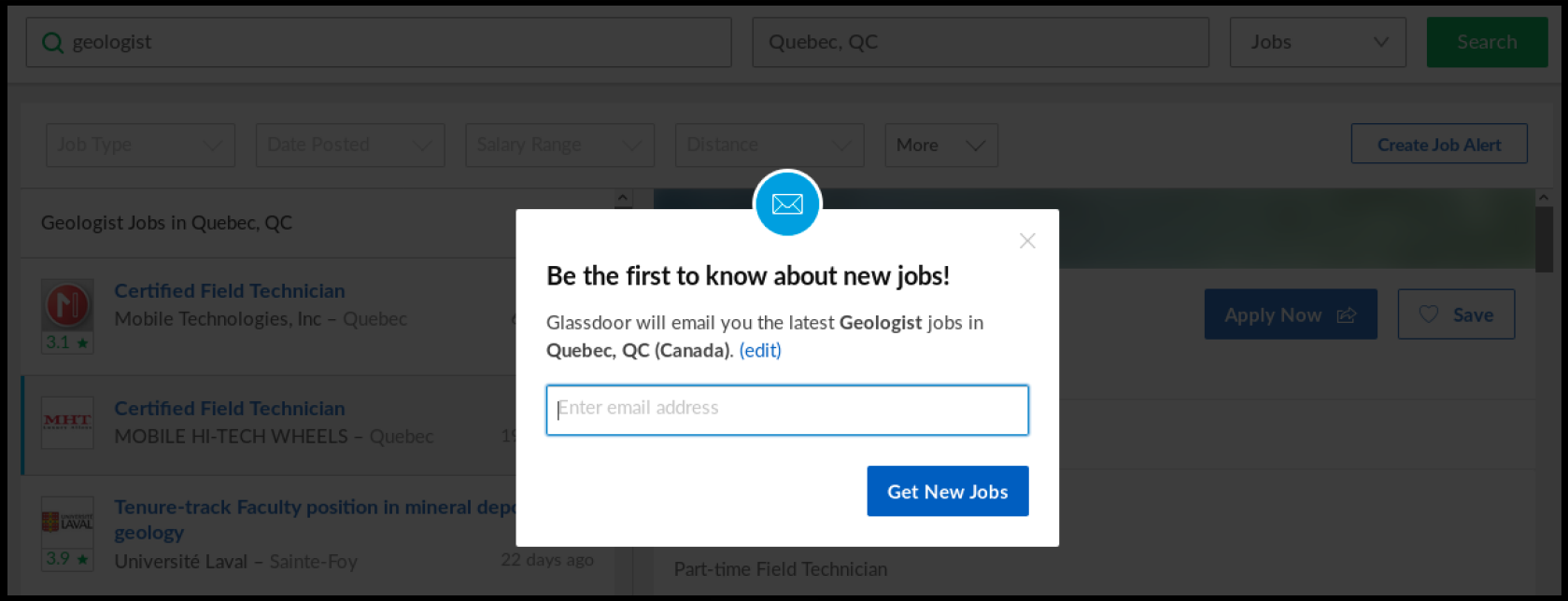
Timing

- Suppose we're implementing a tool like *httpscreenshot*
  - Load every page in a list and screenshot it
- What do we mean by *load*?
  - "Time-bomb" scripts
  - Ultimately, the halting problem

Timing example



Timing example



Implicit waits

```
wd.implicitly_wait(30)
wd.find_element_by_id("username").send_keys(username)
wd.find_element_by_id("password").send_keys(password)
```

- Implicit wait (set once) polls the DOM until the desired element exists
- That is frequently not good enough!
- What can go wrong with the above?

Implicit waits

```
wd.implicitly_wait(30)
wd.find_element_by_id("username").send_keys(username)
wd.find_element_by_id("password").send_keys(password)
```

selenium.common.exceptions.ElementNotInteractable
Exception: Message: Element is not visible

?????

Explicit waits

```
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC

wd.find_element_by_id("username").send_keys(username)
WebDriverWait(wd, 30).until(EC.element_to_be_clickable((By.ID, "password")))
wd.find_element_by_id("password").send_keys(password)
```

Composite actions

- Selenium actions vs. user actions
- Listeners can react in subtle ways
  - onmousedown, onmouseover
- Modified DOM can result in stale elements

Composite actions

```
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains

click_me = wd.find_element_by_link_text("Click me!")

action = ActionChains(wd)
action.move_to_element(click_me)
action.click(click_me)

action.perform()
```

selenium.common.exceptions.StaleElementReferenceException

srv/mouseover_and_click.py

Composite actions

```
click_me = wd.find_element_by_link_text("Click me!")
action = ActionChains(wd)
action.move_to_element(click_me)
action.perform()

# Find the changed element
click_me = wd.find_element_by_link_text("Click me!")
click_me.click()
```

Bot detectors

- Captchas, obviously
- Silent alarm detectors like botguard
  - Browser fingerprinting
  - Event listener heuristics
- Different event patterns for mobile clients
- Don't break accessibility

Frames

- Page loaded but elements not available?
- Often a frame issue
- selenium.webdriver.support.expected_conditions.

  frame_to_be_available_and_switch_to_it(locator)
- Don't forget to switch back

PhantomJS

- Another approach to automation
- Selenium and WebDriver *drive* a browser
- PhantomJS *is* a browser
  - Faster, less resource intensive
  - Harder to interact with / easier to detect
  - Can be automated with WebDriver
- Maintainer stepped down in April 2017
  - Headless Chrome cited as likely replacement

Headless Selenium

- Browsers increasingly support native headless mode
- Firefox and Chrome very similar

```
from selenium import webdriver


options = webdriver.chrome.options.Options()
options.add_argument("--headless")

wd = webdriver.Chrome(chrome_options=options)

wd.implicitly_wait(30)
wd.get("http://localhost:8000")
wd.get_screenshot_as_file("wiki.png")
```

screenshot_chrome.py

Headless Selenium

- Browsers increasingly support native headless mode
- Firefox and Chrome very similar

```
from selenium import webdriver


options = webdriver.firefox.options.Options()
options.add_argument("--headless")

wd = webdriver.Firefox(firefox_options=options)

wd.implicitly_wait(30)
wd.get("http://localhost:8000")
wd.get_screenshot_as_file("wiki.png")
```

screenshot_firefox.py

Headless Selenium

- xvfb is a fallback option
  - xvfb-run <selenium script>
  - pyvirtualdisplay

```
from selenium import webdriver
from pyvirtualdisplay import Display


Display(visible=0).start()

wd = webdriver.Firefox()
wd.get("http://localhost:8000")
wd.get_screenshot_as_file("wiki.png")
```

When not to use Selenium

- Whenever you don't have to!
- Not an API replacement
- Impact to non-Selenium users by blacklisting IPs or associated accounts

References and further reading

- **WebDriver (W3C Candidate Recommendation)**
  - https://www.w3.org/TR/webdriver/
- **Shape Security Blog: Detecting PhantomJS Based Visitors**
  - https://blog.shapesecurity.com/2015/01/22/detecting-phantomjs-based-visitors/
- **Running Selenium with Headless Chrome**
  - https://intoli.com/blog/running-selenium-with-headless-chrome/
- **[Announcement] Stepping down as maintainer**
  - https://groups.google.com/forum/#!msg/phantomjs/9aI5d-LDuNE/5Z3SMZrqAQAJ

Contact

- reiser@defensivecomputing.io

- https://github.com/adamreiser/selenium