PARSNIP

Lightweight Macro Recording and Playback for Google chrome

Shuyang Li, Timothy Seah, James Wang, David Zhao

Motivation

There are many online tasks that are **repetitive**, tedious, or otherwise lend themselves well to automation.

PARSNIP

Motivation

There are many online tasks that are repetitive, **tedious**, or otherwise lend themselves well to automation.

Motivation

There are many online tasks that are repetitive, tedious, or otherwise lend themselves well to **automation.**

Motivation

There are many online tasks that are repetitive, tedious, or otherwise lend themselves well to automation.

Examples:

- Price monitoring (eBay sniping sites)
- Time-sensitive registration (course enrollment)
- Check status every time interval

A Modest Proposal

A Chrome extension that can:

A Modest Proposal

A Chrome extension that can:

1. **Record** a macro, a series of actions across multiple web pages

A Modest Proposal

A Chrome extension that can:

1. **Record** a macro, a series of actions across multiple web pages
2. **Play** that macro when a condition is met.
    a. Time (e.g. 9pm this Friday), intervals (e.g. every 5 seconds)
    b. Value (e.g. when the price drops below $200)

Previous Related Work

- iMacros for Chrome
- Ad hoc sites (eBay sniping sites, airplane ticket price alerts, etc)

PARSNIP

Use Cases

1) At 9:00am on Sunday, play a YouTube video as an alarm

Use Cases

1) At 9:00am on Sunday, play a Youtube video as an alarm
2) **On your friend's birthday, write a "happy birthday" post on her wall**

Use Cases

1) At 9:00am on Sunday, play a Youtube video as an alarm
2) On your friend's birthday, write a "happy birthday" post on her wall
3) **At 7:30am on April 20th, sign up for courses on TigerHub**

PARSNIP

Use Cases

1) At 9:00am on Sunday, play a YouTube video as an alarm
2) On your friend's birthday, write a "happy birthday" post on her wall
3) At 7:30am on April 20th, sign up for courses on TigerHub
4) **When prices for flight UA 87 drop below $500, receive an email notification and buy the ticket**

Structure: At a glance

Frontend

- JavaScript/HTML/CSS
- Chrome extension

PARSNIP

Structure: At a glance

Frontend

- JavaScript/HTML/CSS
- Chrome extension

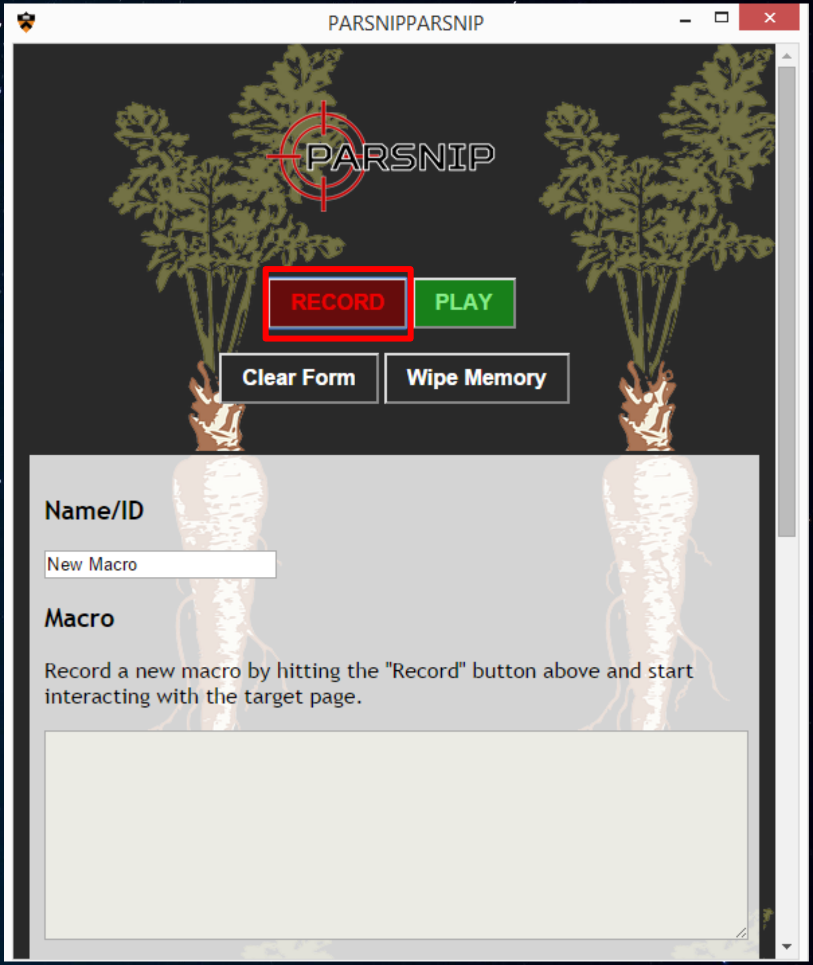
Backend

- Python
- AWS
- SQLite (in progress)

PARSNIP

How it Works: Frontend Pipeline

1. **Inject** JavaScript into every element of page



Name/ID

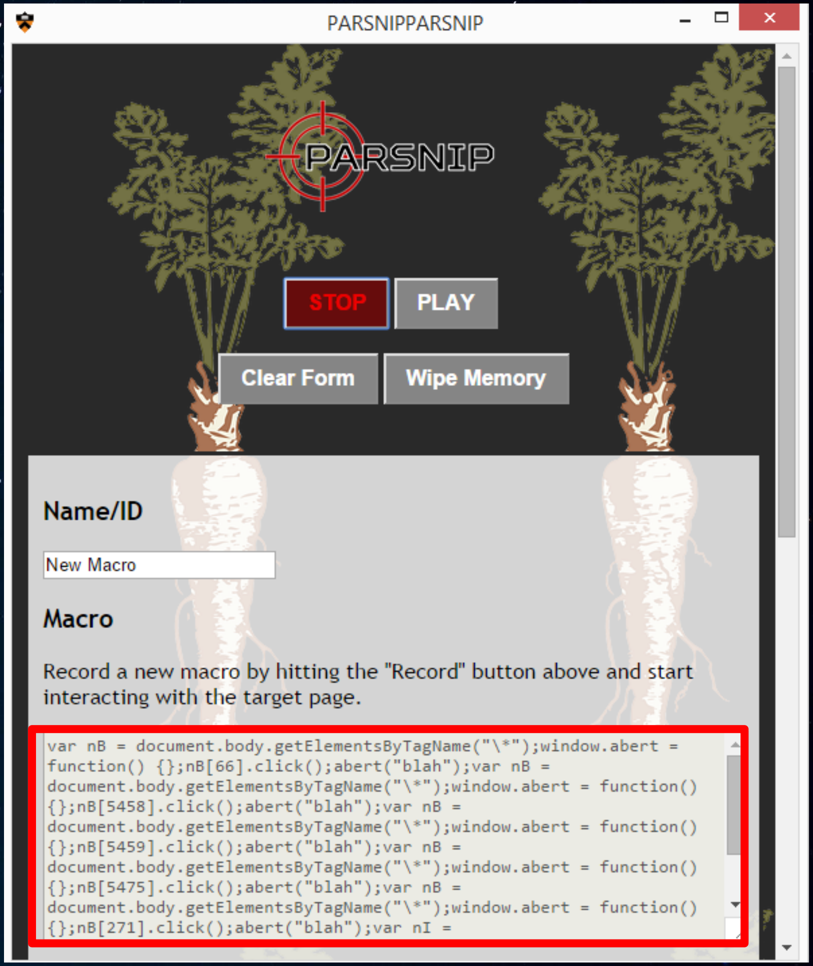
New Macro

Macro

Record a new macro by hitting the "Record" button above and start interacting with the target page.

How it Works: Frontend Pipeline

2. When user **interacts** with element, JavaScript sends code to extension



Name/ID

New Macro

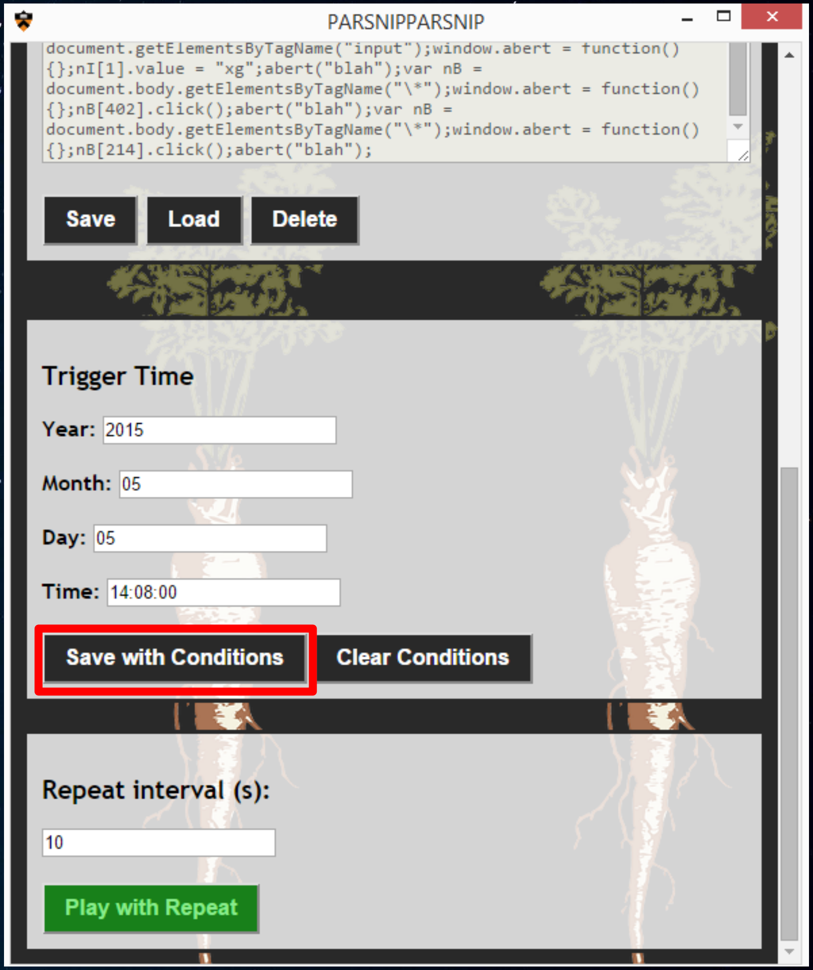
Macro

Record a new macro by hitting the "Record" button above and start interacting with the target page.

```
var nB = document.body.getElementsByTagName("\*");window.abert =
function() {};nB[66].click();abert("blah");var nB =
document.body.getElementsByTagName("\*");window.abert = function()
{};nB[5458].click();abert("blah");var nB =
document.body.getElementsByTagName("\*");window.abert = function()
{};nB[5459].click();abert("blah");var nB =
document.body.getElementsByTagName("\*");window.abert = function()
{};nB[5475].click();abert("blah");var nB =
document.body.getElementsByTagName("\*");window.abert = function()
{};nB[271].click();abert("blah");var nI =
```

How it Works: Frontend Pipeline

3. User (optionally) specifies a **condition**



```
document.getElementsByTagName("input");window.abert = function()
{};nI[1].value = "xg";abert("blah");var nB =
document.body.getElementsByTagName("\*");window.abert = function()
{};nB[402].click();abert("blah");var nB =
document.body.getElementsByTagName("\*");window.abert = function()
{};nB[214].click();abert("blah");
```

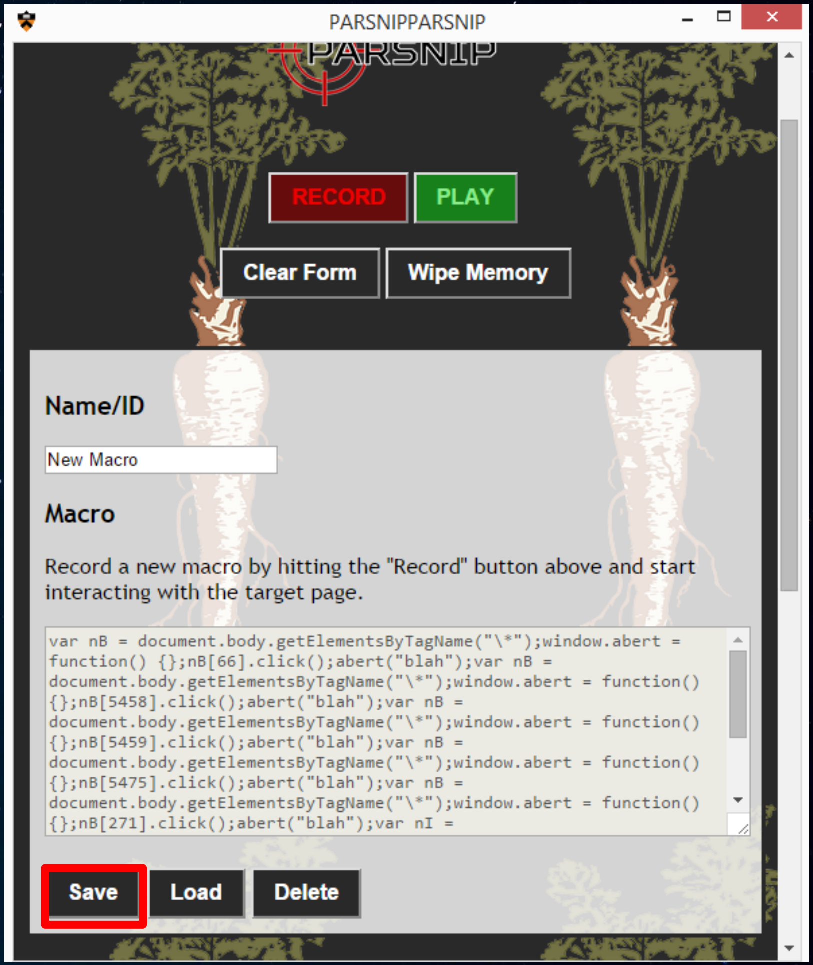
Trigger Time

Year:	2015
Month:	05
Day:	05
Time:	14:08:00

Repeat interval (s): 10

How it Works: Frontend Pipeline

4. User **sends bundled macro** to server using XMLHttpRequest OR **saves locally**



Name/ID

New Macro

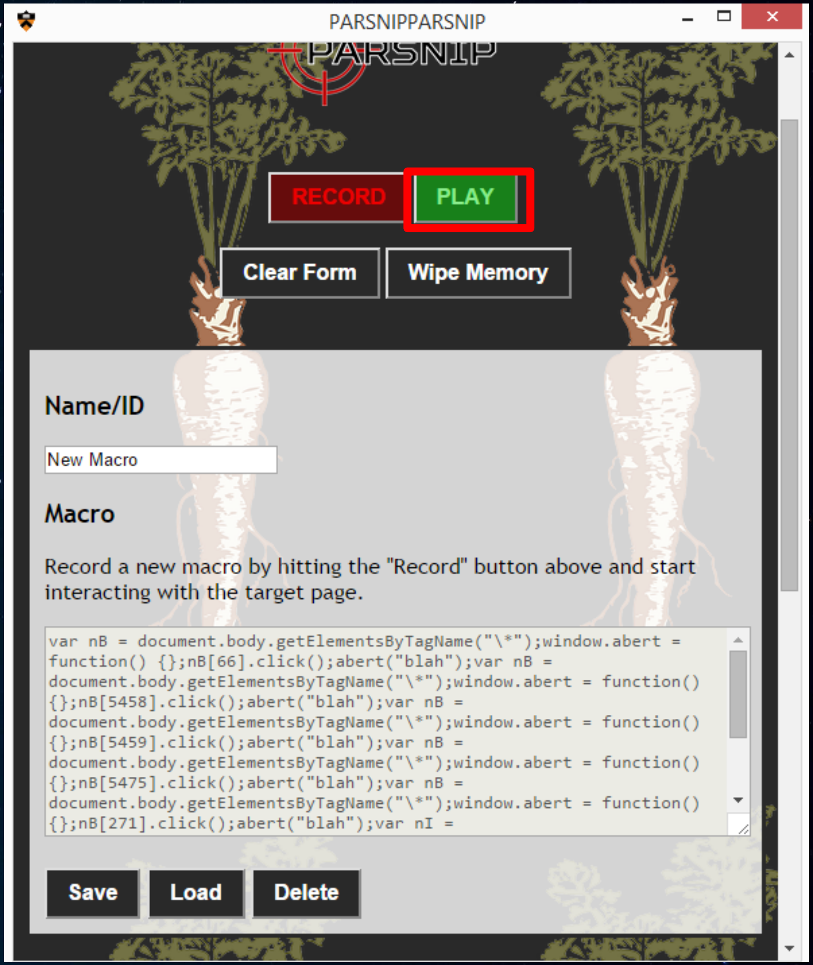
Macro

Record a new macro by hitting the "Record" button above and start interacting with the target page.

```
var nB = document.body.getElementsByTagName("\*");window.abert =
function() {};nB[66].click();abert("blah");var nB =
document.body.getElementsByTagName("\*");window.abert = function()
{};nB[5458].click();abert("blah");var nB =
document.body.getElementsByTagName("\*");window.abert = function()
{};nB[5459].click();abert("blah");var nB =
document.body.getElementsByTagName("\*");window.abert = function()
{};nB[5475].click();abert("blah");var nB =
document.body.getElementsByTagName("\*");window.abert = function()
{};nB[271].click();abert("blah");var nI =
```

How it Works: Frontend Pipeline

5. User **executes** the macro (with or without conditions)



Name/ID

New Macro

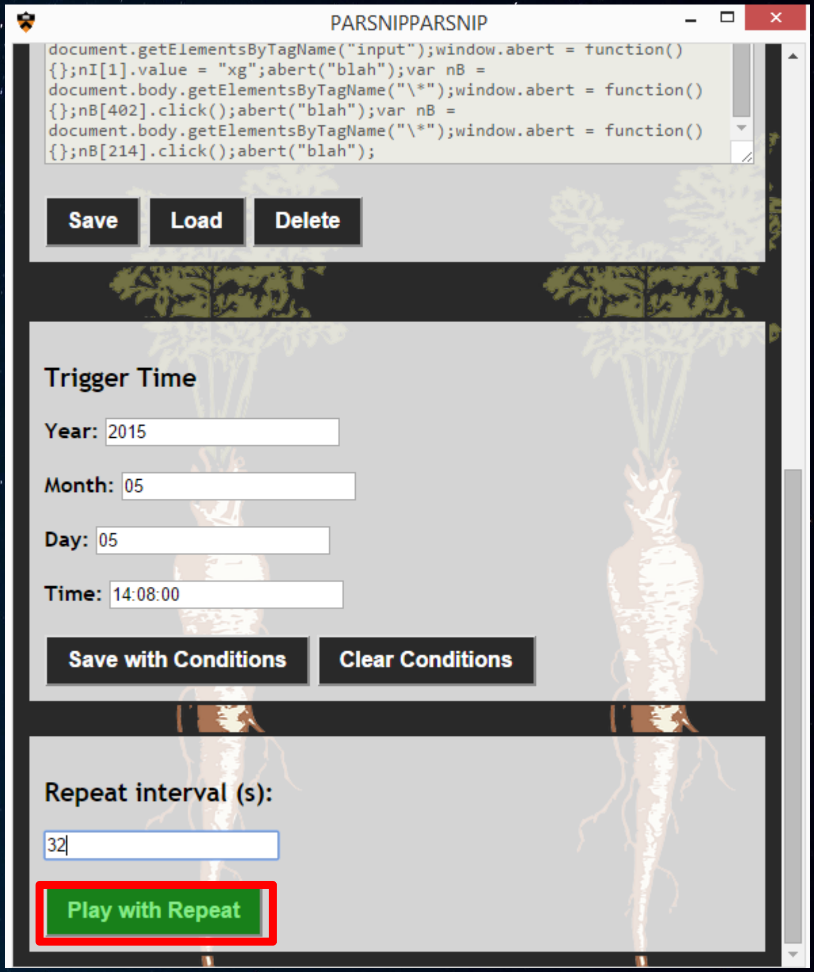
Macro

Record a new macro by hitting the "Record" button above and start interacting with the target page.

```
var nB = document.body.getElementsByTagName("\*");window.abert =
function() {};nB[66].click();abert("blah");var nB =
document.body.getElementsByTagName("\*");window.abert = function()
{};nB[5458].click();abert("blah");var nB =
document.body.getElementsByTagName("\*");window.abert = function()
{};nB[5459].click();abert("blah");var nB =
document.body.getElementsByTagName("\*");window.abert = function()
{};nB[5475].click();abert("blah");var nB =
document.body.getElementsByTagName("\*");window.abert = function()
{};nB[271].click();abert("blah");var nI =
```

How it Works: Frontend Pipeline

6. User can repeatedly play the macro at set **intervals**



```
document.getElementsByTagName("input");window.abert = function()
{};nI[1].value = "xg";abert("blah");var nB =
document.body.getElementsByTagName("\*");window.abert = function()
{};nB[402].click();abert("blah");var nB =
document.body.getElementsByTagName("\*");window.abert = function()
{};nB[214].click();abert("blah");
```

Trigger Time

Year: 2015

Month: 05

Day: 05

Time: 14:08:00

Repeat interval (s): 32

How it Works: Backend Pipeline

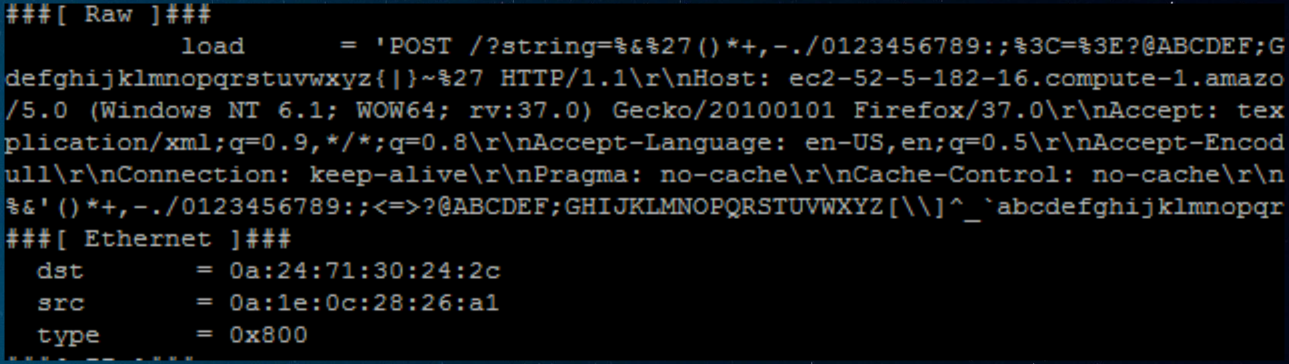
1. Python packet sniffer script

2. SQLite3 Database

3. Splinter + Headless Browser

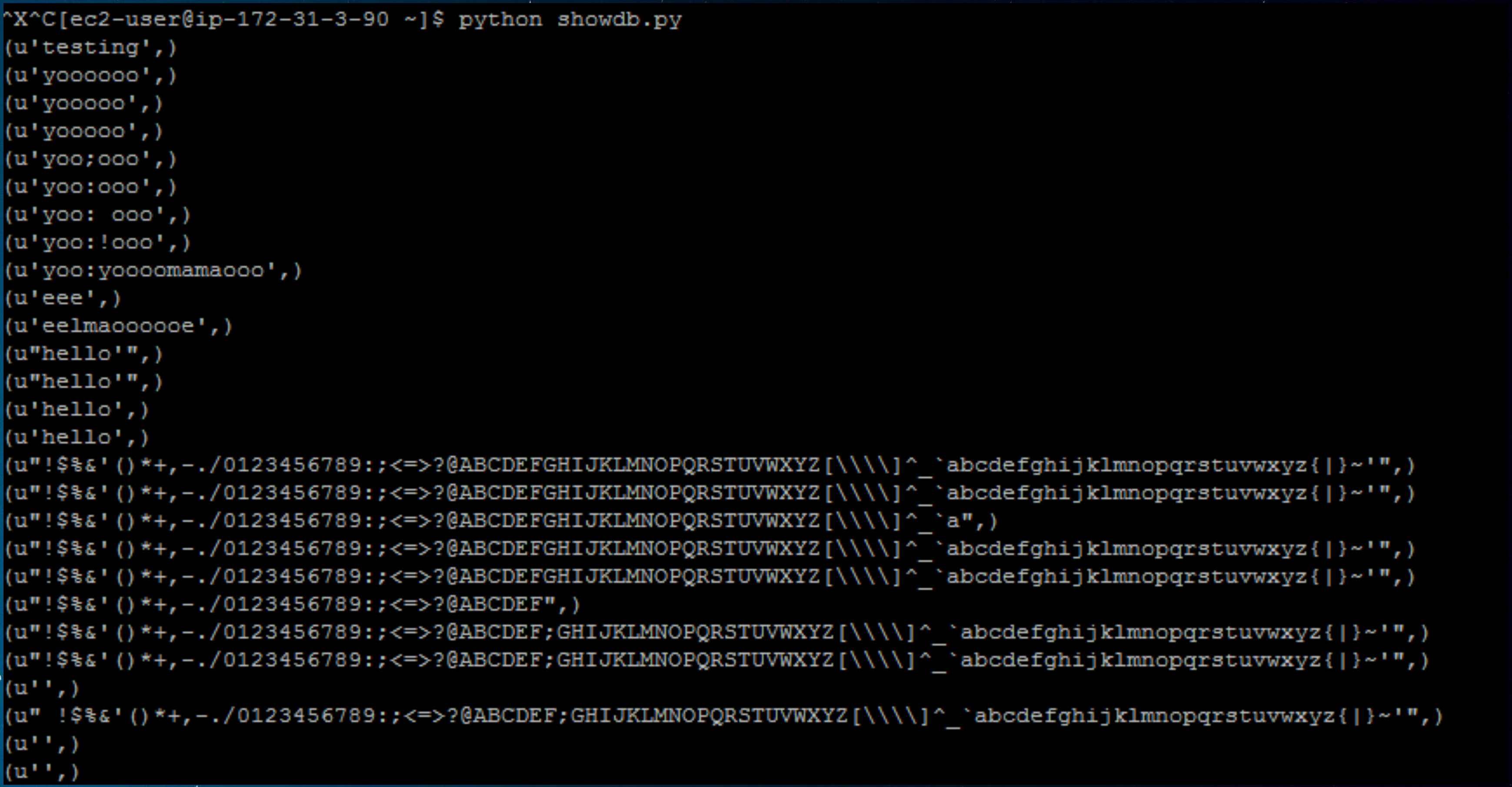
Backend: Python sniffer script

- Uses the **Scapy** package to process incoming packets



```
###[ Raw ]###
           load      = 'POST /?string=%&%27()*+,-./0123456789:;%3C=%3E?@ABCDEF;G
defghijklmnopqrstuvwxyz{|}~%27 HTTP/1.1\r\nHost: ec2-52-5-182-16.compute-1.amazo
/5.0 (Windows NT 6.1; WOW64; rv:37.0) Gecko/20100101 Firefox/37.0\r\nAccept: tex
plication/xml;q=0.9,*/*;q=0.8\r\nAccept-Language: en-US,en;q=0.5\r\nAccept-Encod
ull\r\nConnection: keep-alive\r\nPragma: no-cache\r\nCache-Control: no-cache\r\n
%&'()*+,-./0123456789:;<=>?@ABCDEF;GHIJKLMNOPQRSTUVWXYZ[\\]^_`abcdefghijklmnopqr
###[ Ethernet ]###
  dst       = 0a:24:71:30:24:2c
  src       = 0a:1e:0c:28:26:a1
  type      = 0x800
```

Backend: SQLite3



```
^X^C[ec2-user@ip-172-31-3-90 ~]$ python showdb.py
(u'testing',)
(u'yoooooo',)
(u'yooooo',)
(u'yooooo',)
(u'yoo;ooo',)
(u'yoo:ooo',)
(u'yoo: ooo',)
(u'yoo:!ooo',)
(u'yoo:yoooomamaooo',)
(u'eee',)
(u'eelmaoooooe',)
(u"hello'",)
(u"hello'",)
(u'hello',)
(u'hello',)
(u"!$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\\\]^_`abcdefghijklmnopqrstuvwxyz{|}~'",)
(u"!$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\\\]^_`abcdefghijklmnopqrstuvwxyz{|}~'",)
(u"!$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\\\]^_`a",)
(u"!$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\\\]^_`abcdefghijklmnopqrstuvwxyz{|}~'",)
(u"!$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\\\]^_`abcdefghijklmnopqrstuvwxyz{|}~'",)
(u"!$%&'()*+,-./0123456789:;<=>?@ABCDEF",)
(u"!$%&'()*+,-./0123456789:;<=>?@ABCDEF;GHIJKLMNOPQRSTUVWXYZ[\\\\]^_`abcdefghijklmnopqrstuvwxyz{|}~'",)
(u"!$%&'()*+,-./0123456789:;<=>?@ABCDEF;GHIJKLMNOPQRSTUVWXYZ[\\\\]^_`abcdefghijklmnopqrstuvwxyz{|}~'",)
(u'',)
(u" !$%&'()*+,-./0123456789:;<=>?@ABCDEF;GHIJKLMNOPQRSTUVWXYZ[\\\\]^_`abcdefghijklmnopqrstuvwxyz{|}~'",)
(u'',)
(u'',)
```

Backend: Headless browser

Our browser of choice: **PhantomJS**

- Alternative to executing locally
- Example:
  - Local computer is asleep or turned off

Testing Process

1. 2 types of tests
    a. Macro functionality: link clicks, form submits, text entry, etc
    b. Extension functionality: pressing buttons in different orders

2. Have each group member test each case independently
    a. Report findings on shared Google doc

Known Issues

1. Cannot record macros for all pages - tricky HTML stuff
2. Cannot interact with pages that don't exist yet e.g. "Confirm" on TigerHub
3. Sometimes don't want to execute last step (e.g. buying a plane ticket)
4. Recording macros while logged in

Future Work

1. Record more types of actions!
2. Add "value conditions": match JavaScript element on page e.g. stock prices, ticket fares, etc
3. Backend Execution
4. Add option to record a simulated click (e.g. Ctrl + click)

PARSNIP