

WEB 2.0 启发式爬虫实战

猪猪侠 / 2018年06月16日

关于我

- 阿里云高级安全专家
- 十一年安全从业经历
- 信息安全领域爱好者
 - 自动化安全测试
 - 数据挖掘
- 微博: @ringzero

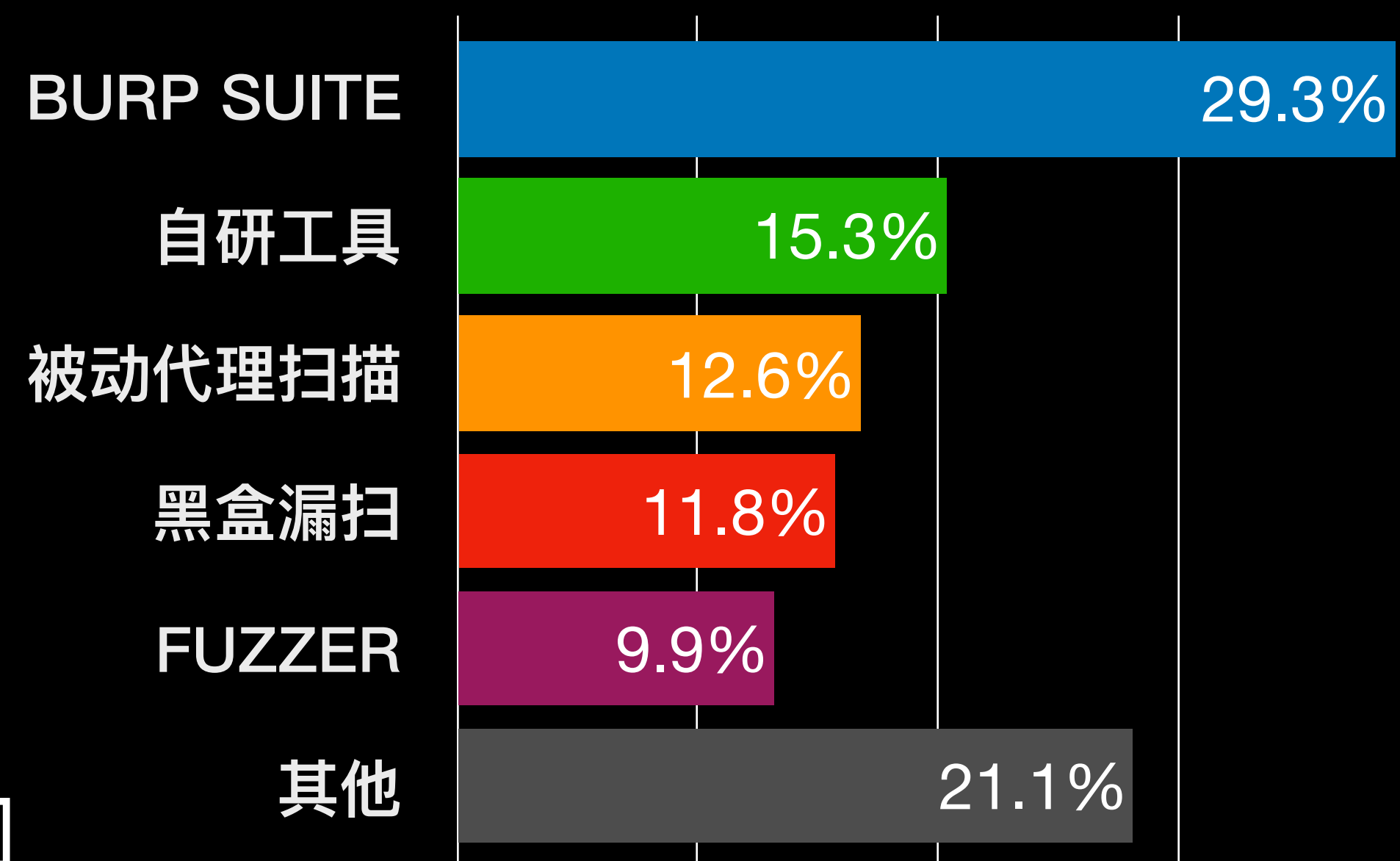


为什么我们需要一个扫描器爬虫？

- 1 安全测试自动化程度低（人工时代）
- 2 大量的人肉测试重复成本投入
- 3 被测试系统攻击面被遗漏
- 4 安全测试用例被遗漏
- 5 WEB 2.0 前端框架导致复杂度增加

Angular、React、Vue

测试工具使用情况



现在业界是如何实现爬虫的?

测试目标

`curl http://www.seebug.org`

网站特征

- Vue.js
- JQuery
- Handlebars
- 代码混淆反爬虫
- DOM 事件频繁更新



正则大法无解

静态链接分析: **无结果**

BeautifulSoup4 / HTMLParser

```
#!/usr/bin/env python3
# encoding: utf-8

import requests
from bs4 import BeautifulSoup

resp = requests.get('https://www.seebug.org/')
soup = BeautifulSoup(resp.content, 'lxml')
resources = {
    'anchor': (soup.find_all('a'), 'href'),
    'iframe': (soup.find_all('iframe'), 'src'),
    'frame': (soup.find_all('frame'), 'src'),
    'img': (soup.find_all('img'), 'src'),
    'link': (soup.find_all('link'), 'href'),
    'script': (soup.find_all('script'), 'src'),
    'form': (soup.find_all('form'), 'action'),
}

print(resources)
```

WEB 2.0 动态爬虫应运而生：基于无界面浏览器

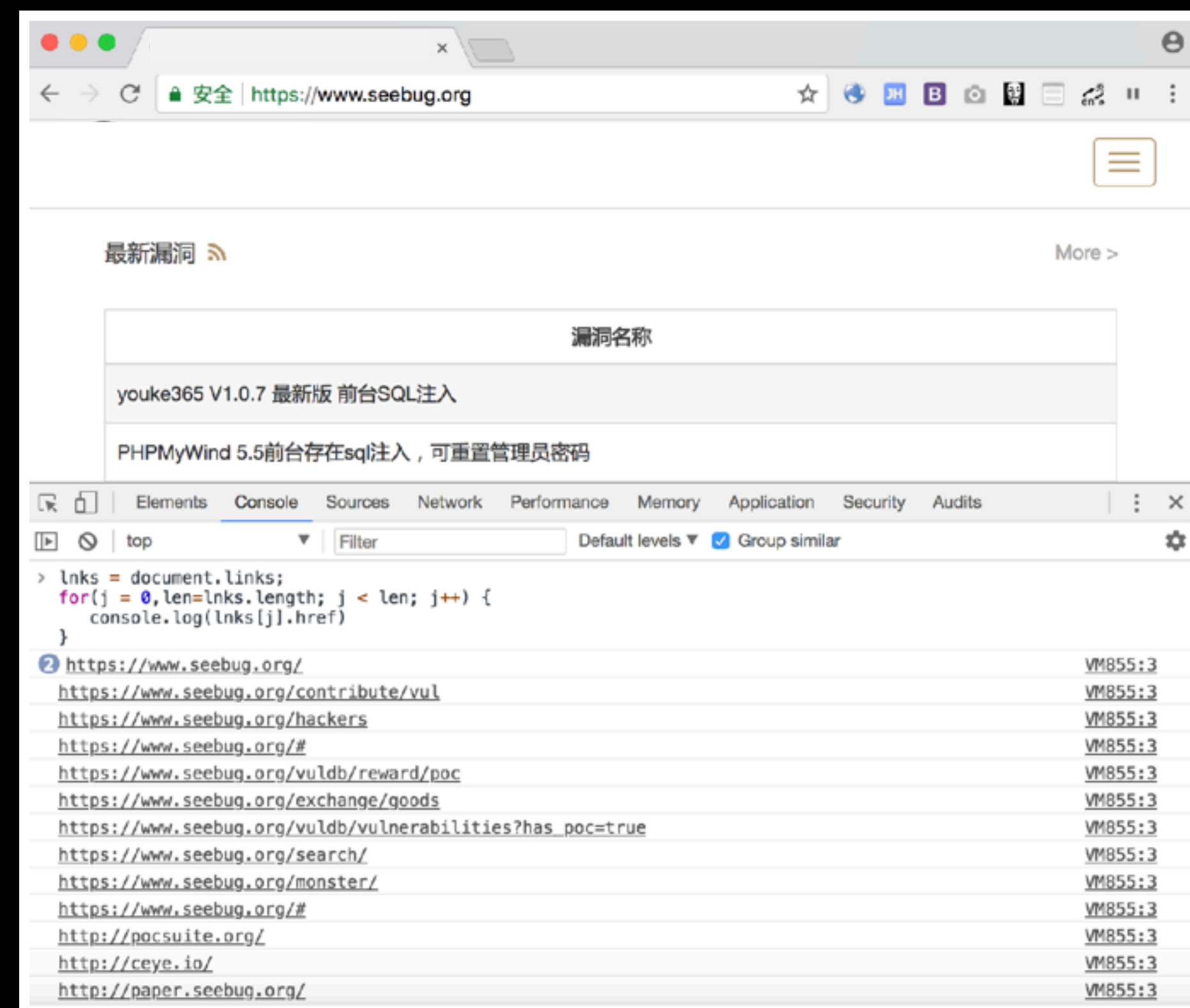
```
$pip3 install pyppeteer  
https://github.com/miyakogi/pyppeteer
```

```
#!/usr/bin/env python3  
# encoding: utf-8  
  
import asyncio  
from pyppeteer import launch  
  
async def main():  
    browser = await launch()  
    page = await browser.newPage()  
    await page.goto('http://www.seebug.org')  
    await page.waitFor("body > div.footer-up")  
  
    urls = await page.evaluate('''() => {  
        var urls = new Array();  
        var atags = document.getElementsByTagName("a");  
        for(var i=0;i<atags.length;i++){  
            if (atags[i].getAttribute("href")){  
                urls[i] = atags[i].getAttribute("href")  
            }  
        }  
        return urls;  
    }''')  
  
    print(urls)  
    await browser.close()  
  
asyncio.get_event_loop().run_until_complete(main())
```




Chromium

puppeteer

pyppeteer



无界面浏览器历史

	QtWebkit	Ghost.py	PyQt4
	PhantomJS	CasperJS	SlimmerJS
	Chromium Headless		

为什么选择 Chromium Headless



- 谷歌大厂支持 市场第一 几小时一个版本更新
- 积极支持 W3C标准组织

浏览器	Chrome	Firefox	Internet Explorer	Safari	Opera
渲染引擎	Blink	Gecko	Trident	WebKit	Blink
JS引擎	V8	OdinMonkey	Chakra	JSCore	Carakan

HTML	CSS	DOM	XPath	JavaScript
HTML 5	CSS3	DOM4	XPath 2.0	AJAX
XHTML 1.0	CSS2	DOM3	XQuery 1.0	ECMAScript 5.1
HTML 4.01	CSS1	DOM Events	XPath 1.0	ECMAScript 6
		DOM Core		

如何安装 Chromium Headless 结合 SLB 实现微服务化

1 环境需求: CentOS 7 (Google被墙)

2



```
yum install -y ipa-gothic-fonts xorg-x11-fonts-100dpi xorg-x11-fonts-75dpi xorg-x11-utils  
xorg-x11-fonts-cyrillic xorg-x11-fonts-Type1 xorg-x11-fonts-misc pango.x86_64  
libXcomposite.x86_64 libXcursor.x86_64 libXdamage.x86_64 libXext.x86_64 libXi.x86_64  
libXtst.x86_64 cups-libs.x86_64 libXScrnSaver.x86_64 libXrandr.x86_64 GConf2.x86_64 alsa-  
lib.x86_64 atk.x86_64 gtk3.x86_64
```

3

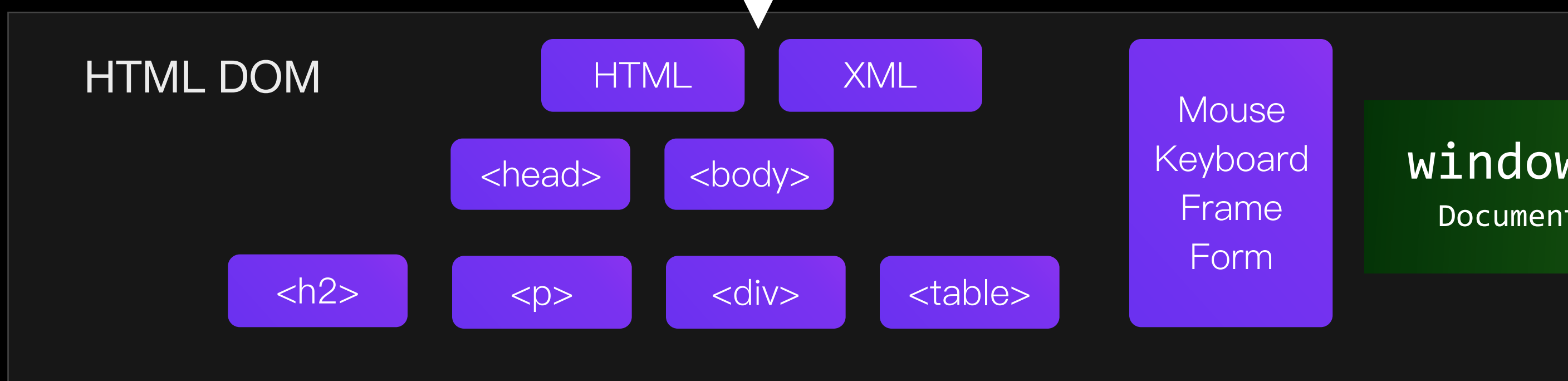
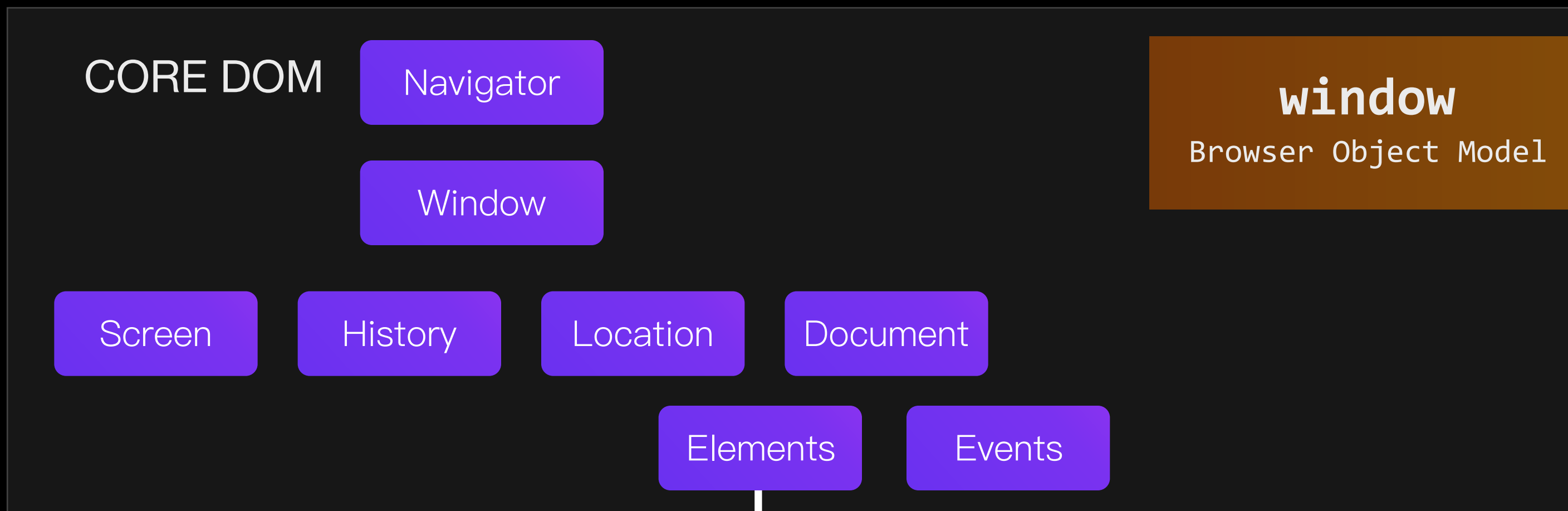


```
$ git clone https://github.com/scheib/chromium-latest-linux  
$ cd chromium-latest-linux  
$ ./update.sh  
$ nohup ./latest/chrome --headless --disable-gpu --remote-debugging-port=9222  
--remote-debugging-address=0.0.0.0 --disable-web-security --disable-xss-auditor  
--no-sandbox --disable-setuid-sandbox &
```

Chrome Headless 命令行参数 选项 列表 <https://peter.sh/experiments/chromium-command-line-switches/>

4 DevTools listening on web socket http://{server}:9222/json/version

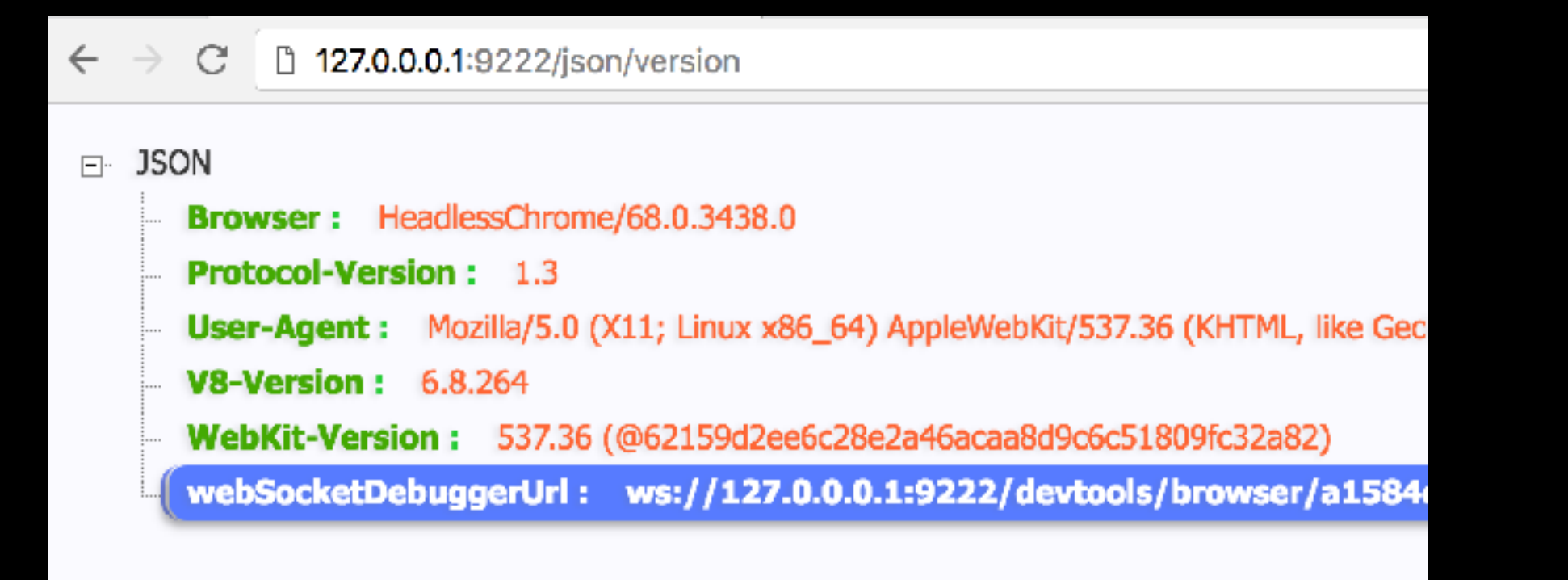
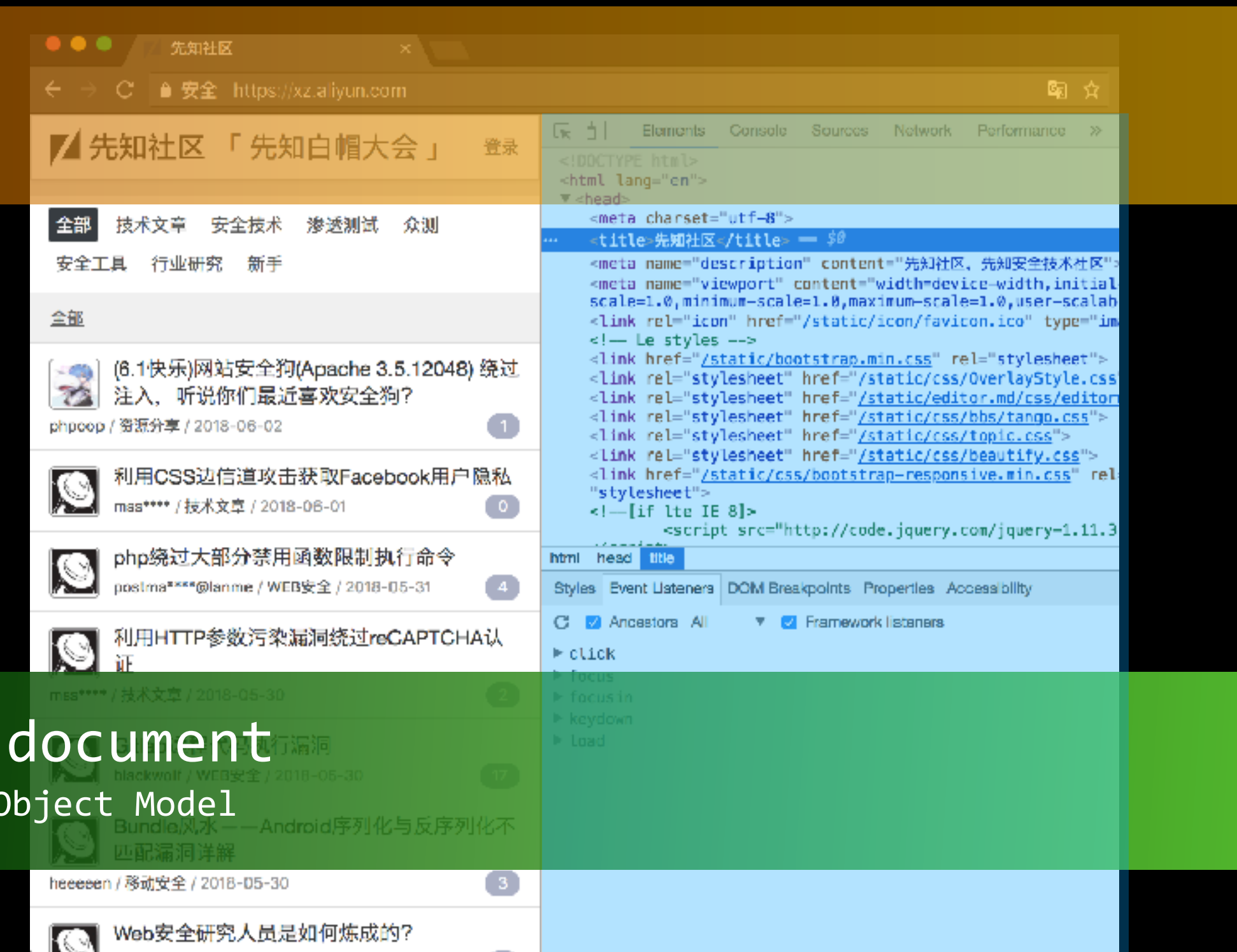
WEB 2.0 爬虫实践基础 BOM | DOM | CDP



```
import asyncio
from pyppeteer import launch

async def main():
    browser = await connect({'browserWSEndpoint': '{websocketDebuggerUrl}'})
    page = await browser.newPage()
    await page.goto('https://xz.aliyun.com/')
    asyncio.get_event_loop().run_until_complete(main())
```

CDP
Chrome DevTools Protocol



websocketDebuggerUrl : http://127.0.0.1:9222/json/version

CDP – Chrome DevTools Protocol 原理剖析

1 打开主进程，启动CDP Socket Server服务

2 创建 websocket 连接 DevTools

```
websocket_url = 'ws://0.0.0.0:9222/devtools'
websocket.create_connection(websocket_url, enable_multithread=True)
```

3 创建一个浏览器的新标签页面

```
command = {
  "method": "Target.createTarget", "params": {u'url': u'about:blank'}
}
```

4 申请一个非共享空间的新标签页面

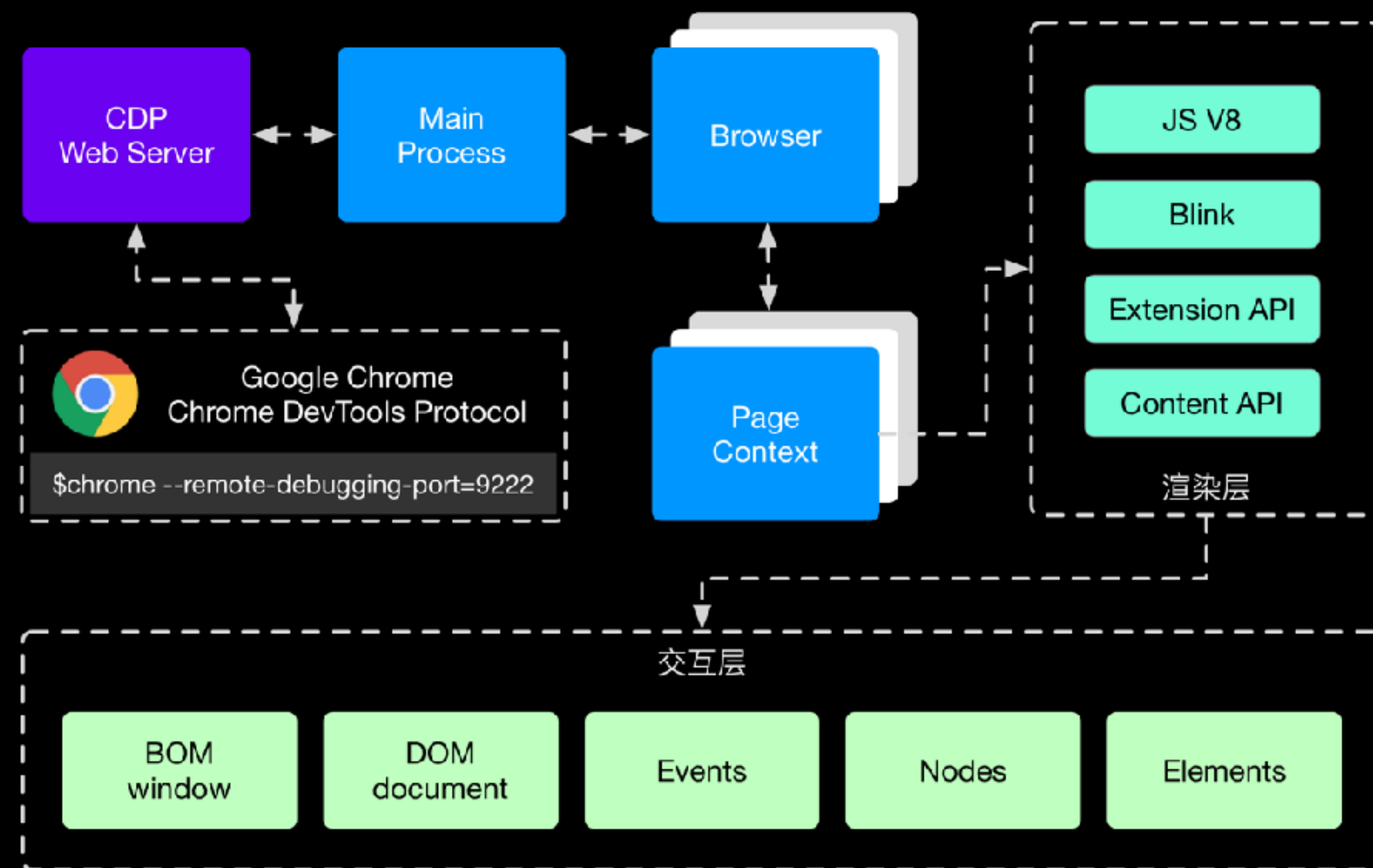
```
command = {"method": "Target.createBrowserContext"}
```

5 利用导航条功能打开特定网站

```
command = {
  "method": "Page.navigate",
  "params": {"url": "https://xz.aliyun.com"}
}
```

6 CDP提供WebShell级别的完美API操控

- Page.getCookies
- Page.captureScreenshot
- Page.printToPDF



pyppeteer

Chrome DevTools Protocol Viewer

<https://chromedevtools.github.io/devtools-protocol/tot/Page>

Chromium 初体验：精准便捷的挖掘反射/DOM XSS漏洞

http://210.158.41.67/MCIR/xssmh/xss.php?inject_string=payload

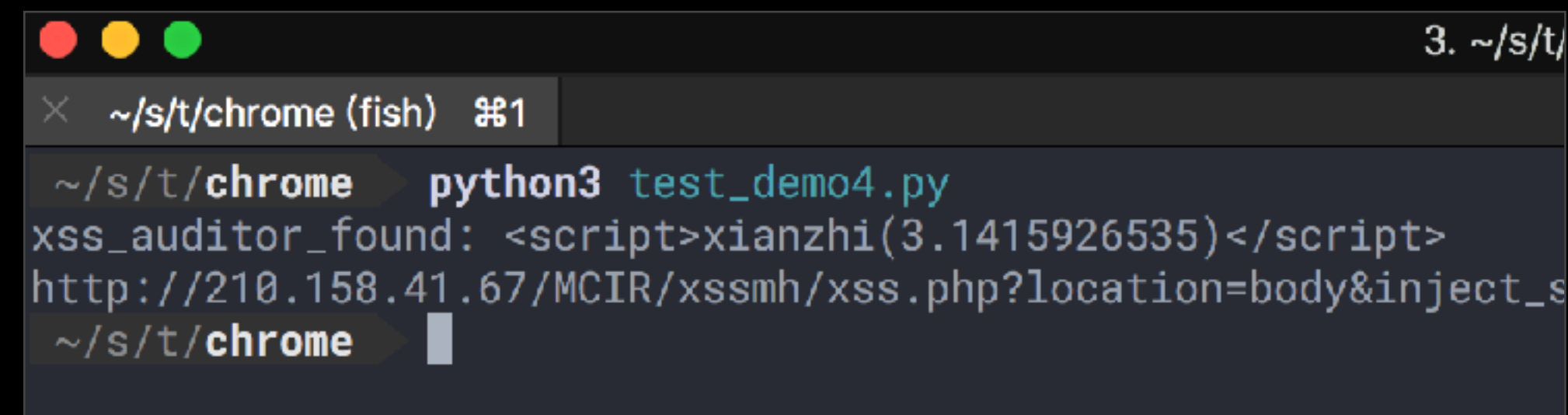
OWASP Broken Web Applications Project

PAYLOAD LIST

- ① `<script>xianzhi(3.1415926535)</script>`
- ② `<xianzhi></xianzhi>`
- ③ `test"onmouseover=xianzhi(3.14)"`
- ④ `123"onfocus=xianzhi(3.14) autofocus="`

HOOK 函数验证

- `await page.$('xianzhi') != null`
- `document.getElementsByTagName('xianzhi')`



```
~/s/t/chrome python3 test_demo4.py
xss_auditor_found: <script>xianzhi(3.1415926535)</script>
http://210.158.41.67/MCIR/xssmh/xss.php?location=body&inject_s
~/s/t/chrome
```

```
import asyncio
from pyppeteer import launch

payload = '<script>xianzhi(3.1415926535)</script>'
url = 'http://210.158.41.67/MCIR/xssmh/xss.php?location=body&inject_string={payload}&submit=Inject'.format(payload=payload)

def xss_auditor(url, message):
    if message == 3.1415926535:
        print('xss_auditor_found:', payload)
        print(url)

async def main():

    browser = await launch(headless=False, args=['--disable-xss-auditor'])
    page = await browser.newPage()

    await page.exposeFunction(
        'xianzhi', lambda message: xss_auditor(url, message)
    )

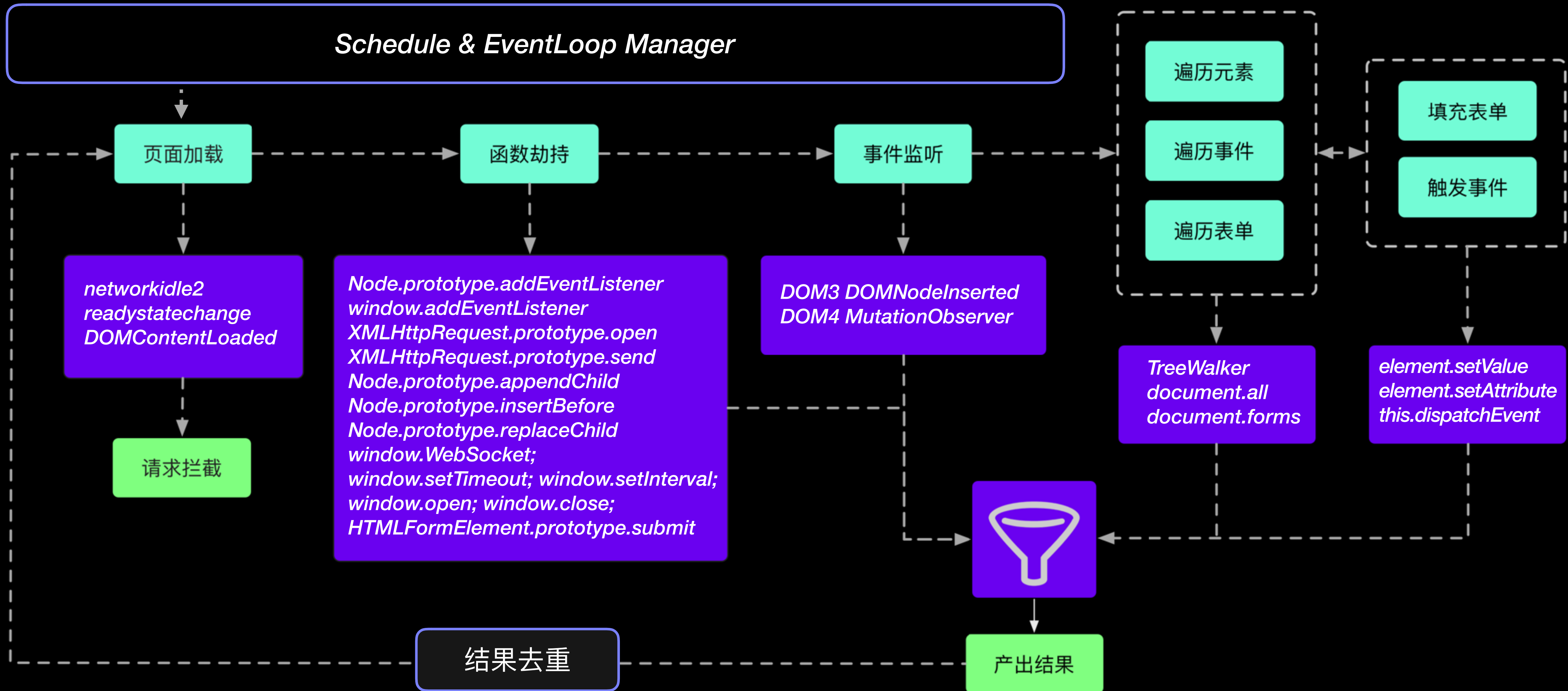
    await page.goto(url)
    await page.close()

asyncio.get_event_loop().run_until_complete(main())
```


什么是启发式爬虫？

“基于历史经验和已知场景，构造并实现规则的爬虫。”

启发式爬虫最佳实践：任务调度及事件管理流程



页面加载：什么时候开始注入代码

- **页面内容加载完成 page load**
- **等待页面加载完成 networkidle2**
- **DOM树解析完成 DOMContentLoaded**

`page.once('load', () => {});`

等待网络状态为空闲的时候才继续执行

初始DOM，加载并解析完成

- `document.readyStatechange`
- `document.readyState === "complete"`

请求拦截：解决页面被意外跳转和关闭

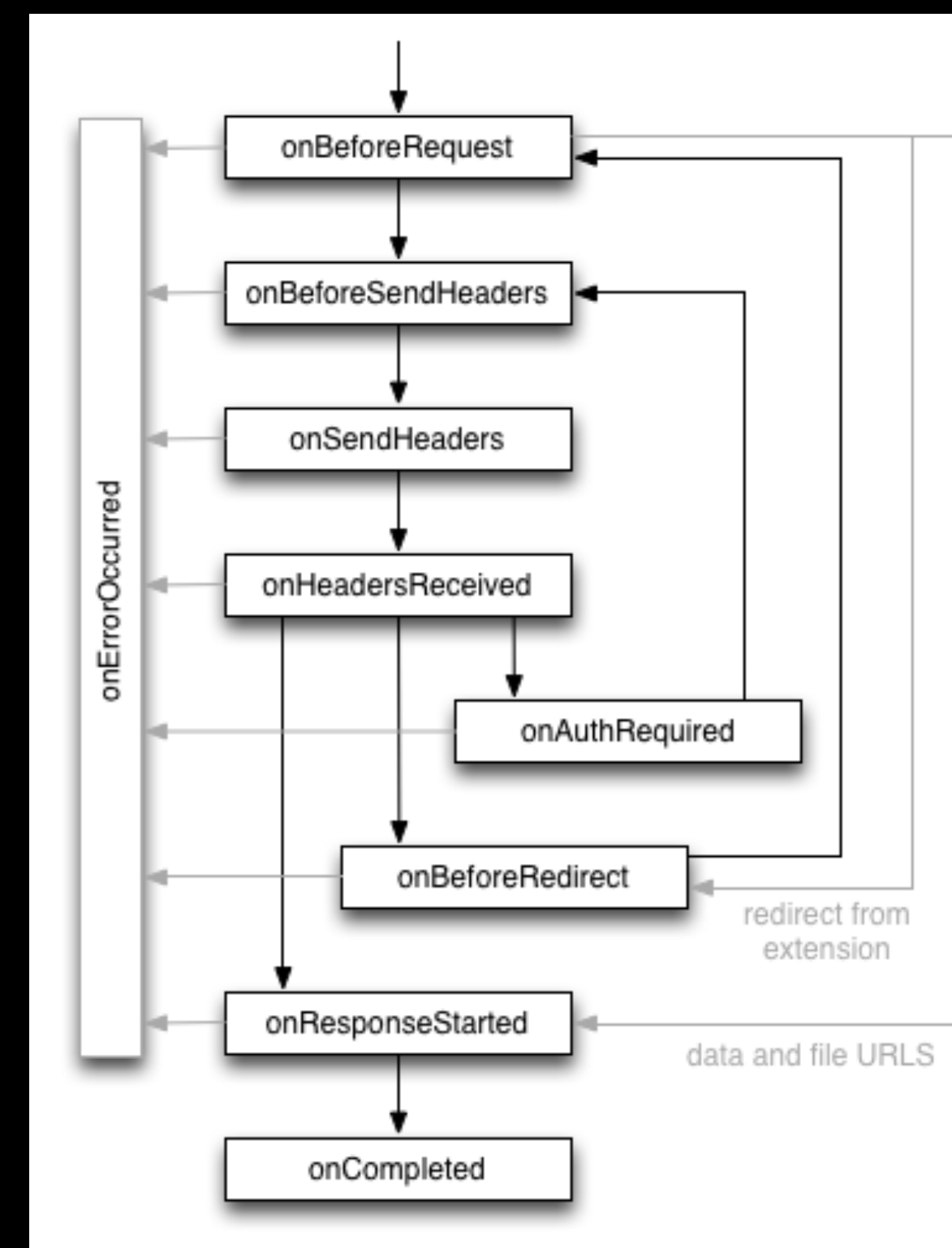
- ① `window.open();`
- ② `window.location = "/123";`
- ③ `window.location = "/456";`

● 插件拦截网络请求

```
chrome.tabs.onCreated.addListener  
chrome.tabs.onUpdated.addListener  
chrome.webRequest.onBeforeRequest.addListener();  
return {redirectUrl: 'javascript:void(0)'};
```

● 启用插件选项 Chromium

- `--load-extension=./xianzhi_ext/`
- `--disable-extensions-except=./xianzhi_ext/`



参考：<https://crxdoc-zh.appspot.com/extensions/webRequest>

函数劫持：页面控制及超时阻塞

- 弹框
- 新页面打开
- 超时阻塞等待

- framenavigated
- alert()
- confirm()
- prompt()
- window.close
- window.setTimeout
- window.setInterval

```
// alert/confirm/prompt
page.on('dialog', dialog => { dialog.accept() });

page.on('framenavigated', frameTo => {
  console.log(frameTo.url());
});

window.close = function() {};

window.open = function(url) { console.log(url); };

window.__originalSetTimeout = window.setTimeout;
window.setTimeout = function() {
  arguments[1] = 0;
  return window.__originalSetTimeout.apply(this, arguments);
};

window.__originalSetInterval = window.setInterval;
window.setInterval = function() {
  arguments[1] = 0;
  return window.__originalSetInterval.apply(this, arguments);
};
```

函数劫持：捕获AJAX的请求信息

● 启用请求拦截过滤处理

```
await page.setRequestInterception(true);
page.on('request', request => {
  console.log(request.url());
  request.continue();
});
```

```
await page.goto('http://demo.aisec.cn/demo/aisec/');
> http://demo.aisec.cn/demo/aisec/
> http://demo.aisec.cn/demo/aisec/ajax_link.php?id=1&t=0.255~
> http://demo.aisec.cn/favicon.ico
```

● 劫持原生类 XMLHttpRequest

```
XMLHttpRequest.prototype.__originalOpen = XMLHttpRequest.prototype.open;
XMLHttpRequest.prototype.open = function(method, url, async, user, password) {
  // hook code
  return this.__originalOpen(method, url, async, user, password);
}
```

```
XMLHttpRequest.prototype.__originalSend = XMLHttpRequest.prototype.send;
XMLHttpRequest.prototype.send = function(data) {
  // hook code
  return this.__originalSend(data);
}
```


事件监听：获取新增绑定事件变更信息

TEST

“JavaScript中绑定事件，
均需要调用
addEventListener函数。”

W3C DOM 规范中提供的注册事件监听器

addEventListener

```
// <button id="y">TEST</button>

y.addEventListener('click', function (element) {
  console.log(element);
}, false);
```

```
_addEventListener = Element.prototype.addEventListener;
Element.prototype.addEventListener = function() {
  console.log(arguments, this)
  _addEventListener.apply(this, arguments);
};

window.__originalAddEventListener = window.addEventListener;
window.addEventListener = function() {
  console.log(arguments, this)
  window.__originalAddEventListener.apply(this, arguments);
};
```

事件监听：获取事件被触发后的节点属性变更信息

“DOM4新增的 MutationObserver方法，可监控页面中节点属性发生改变时的细节。”

W3C DOM4

DOMNodeInserted

W3C DOM3

```
var observer = new WebKitMutationObserver(function(mutations ){
  console.log('eventLoop nodesMutated:', mutations.length);
  mutations.forEach(function (mutation) {
    if (mutation.type === 'childList') {
      for (let i = 0; i < mutation.addedNodes.length; i++) {
        let addedNode = mutation.addedNodes[i];
        console.log('Node added:', addedNode.nodeType, mutation.addedNodes[i]);
      }
    } else if (mutation.type === 'attributes') {
      let element = mutation.target;
      var element_val = element.getAttribute(mutation.attributeName)
      console.log(mutation.attributeName, '->', element_val)
    }
  });
});

observer.observe(window.document.documentElement, {
  childList: true,
  attributes: true,
  characterData: false,
  subtree: true,
  characterDataOldValue: false,
  attributeFilter: ['src', 'href'],
});
```

遍历节点及事件：DOM节点属性和绑定事件

`document.all`

`document.createTreeWalker`

“页面加载完成后，渲染引擎完成DOM、CSSOM的渲染，网页所有节点元素及事件注册就绪。”

链接信息

```
var lns = ['src', 'href', 'action']  
lns.includes(attrs[k].nodeName)
```

```
nodes = document.all;  
for(j = 0; j < nodes.length; j++) {  
  attrs = nodes[j].attributes;  
  for(k=0; k<attrs.length; k++) {  
    if (attrs[k].nodeName.startsWith('on')) {  
      console.log(attrs[k].nodeName, attrs[k].nodeValue);  
    }  
  }  
}
```

```
var treeWalker = document.createTreeWalker(  
  document.body,  
  NodeFilter.SHOW_ELEMENT,  
  { acceptNode: function(node) { return NodeFilter.FILTER_ACCEPT; } },  
  false  
);  
  
while(treeWalker.nextNode()) {  
  var element = treeWalker.currentNode  
  console.log(element);  
  if (element.nodeName.startsWith('on')) {  
    console.log(element.nodeName, element.nodeValue);  
  }  
}
```


触发节点上已绑定的事件信息

网页新闻滚动分页

弹框供用户选择选项

后台定时刷新数据

button	select	input	a	textarea	span	td	tr	div
click	change	change	click	change	click	click	click	click
dblclick	click	click	dblclick	click	mouseup	mouseup	mouseup	mouseup
keyup	keyup	blur	keyup	blur	mousedown	mousedown	mousedown	mousedown
keydown	keydown	focus	keydown	focus				scroll
mouseup	mouseup	keyup	mouseup	keyup				
mousedown	mousedown	keydown	mousedown	keydown				
		mouseup		mouseup				
		mousedown		mousedown				

触发节点上已绑定的事件信息

- `dispatchEvent()`
- 简单粗暴执行 `eval()`
- 获取事件类型
- 选择节点对象
- 触发事件

模拟人类的鼠标移动、点击

nodeName = nodeValue

```
// <button id="elem" onclick="alert('Click!');">Click</button>

let event = new Event("click");
elem.dispatchEvent(event);

for(var i=0;i<elem.attributes.length;i++){
  var element = elem.attributes[i]
  if (element.nodeName.startsWith('on')) {
    console.log(element.nodeName);
    eval(element.nodeValue);
  }
}
```

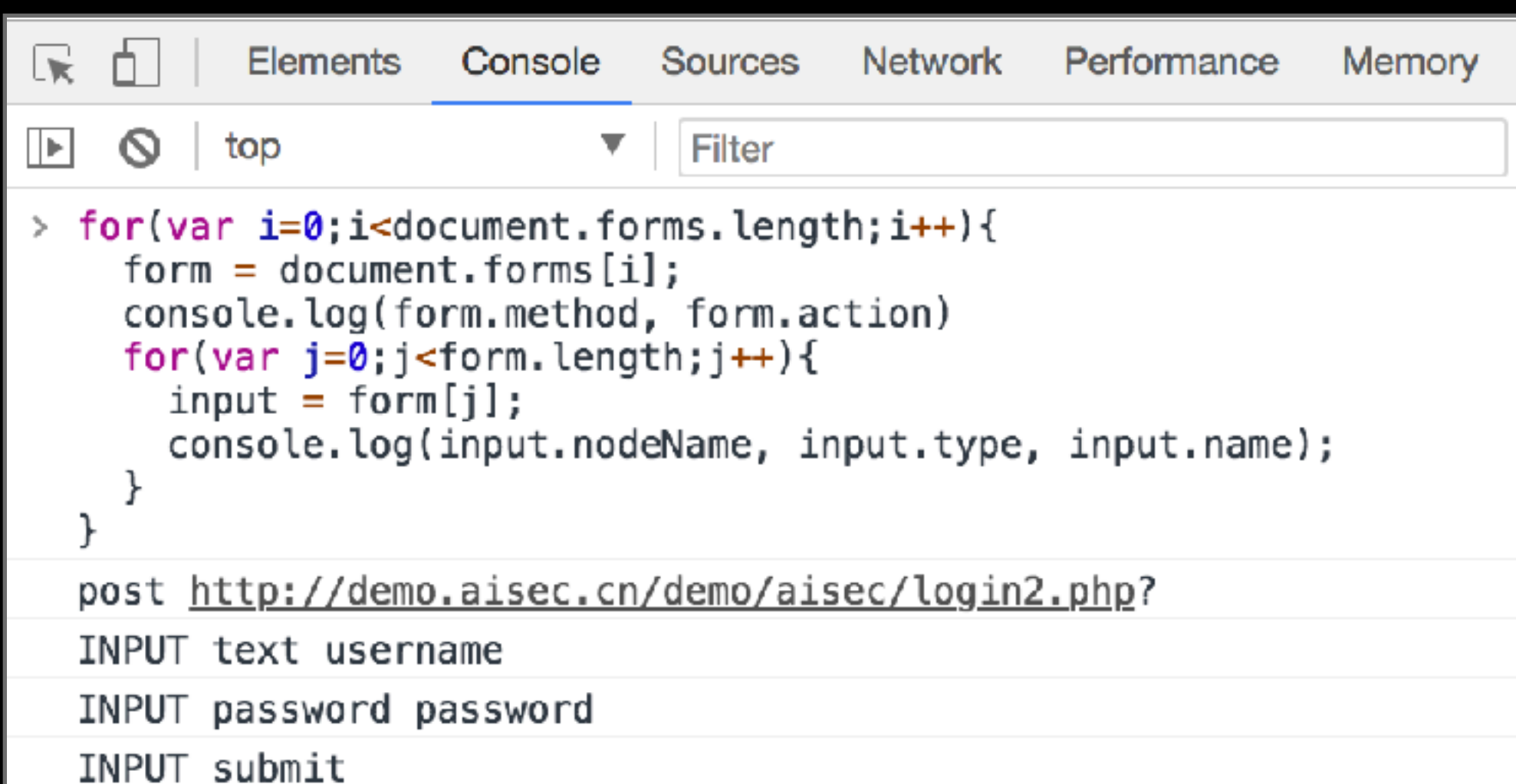
```
var evt = document.createEvent("MouseEvent");
evt.initMouseEvent("click", true, true, window,
  0, 0, 0, 0, 0, false, false, false, false, 0, null);
```

遍历表单：获取参数信息

document.forms

http://demo.aisee.cn/demo/aisee/login2.php

```
for(var i=0;i<document.forms.length;i++){
  form = document.forms[i];
  console.log(form.method, form.action)
  for(var j=0;j<form.length;j++){
    input = form[j];
    console.log(input.nodeName, input.type, input.name);
  }
}
```



```
> for(var i=0;i<document.forms.length;i++){
  form = document.forms[i];
  console.log(form.method, form.action)
  for(var j=0;j<form.length;j++){
    input = form[j];
    console.log(input.nodeName, input.type, input.name);
  }
}
post http://demo.aisee.cn/demo/aisee/login2.php?
INPUT text username
INPUT password password
INPUT submit
```

信息输入

text

search

空名称

静默处理

button

hidden

submit

file

点选框

radio

checkbox

数字区间

range

number

时间

datetime-local

HTML5

password

color

date

email

month

time

url

week

tel

表单参数名称

username

uname

name

password

passwd

pass

email

mail

loginid

填充表单：自动设置参数值

- 自动识别参数长度

max-length
min-length

- 基于名称映射生成组合参数值

element.setValue('{value}')
element.setAttribute('checked')

mail email	{names_en}+{surnames_en}+{random}+{domains}
((number) (phone)) (^tel)	'13'+{random}{9}
(date) (birth)	{years}+{month}+{random}
((month) (day)) (^mon\$)	{years}+{month}+{random}
url website blog	' <u>http://www.</u> '+{random}+{domains}
username uname	{names}+{surnames}

```
letters = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

```
numbers = '0123456789'
```

```
symbols = '!^;.,?%$*#'
```

```
months = ['01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12']
```

```
years = ['1985', '1988', '1990', '1992', '1995', '1996']
```

```
names = ['王', '李', '张', '刘', '陈', '杨', '黄', '周', '赵', '吴', '孙', '徐', '林', '胡', '朱', '郭', '梁', '马', '高', '何']
```

```
surnames = ['娜', '敏', '静', '丽', '强', '磊', '洋', '军', '杰', '芳', '勇', '睿', '宇', '翔', '宥', '品', '彤', '希', '晴']
```

```
names_en = ['wang', 'li', 'zhang', 'liu', 'chen', 'yang', 'huang', 'zhou', 'zhao', 'wu', 'sun', 'xu', 'lin', 'hu', 'zhu', 'guo', 'liang', 'ma', 'gao', 'he']
```

```
surnames_en = ['na', 'min', 'jing', 'li', 'qiang', 'lei', 'yang', 'jun', 'jie', 'fang', 'yong', 'rui', 'yu', 'xiang', 'you', 'pin', 'tong', 'xi', 'qing']
```

```
address = ['广东省广州市', '北京市朝阳区', '浙江省杭州市', '上海市浦东新区']
```

```
domains = ['.com', '.org', '.net', '.cn', '.edu', '.gov.cn']
```


DEMO

下载链接

[https://github.com/ring04h/papers/blob/master/
xianzhi crawler demo.mov](https://github.com/ring04h/papers/blob/master/xianzhi_crawler_demo.mov)

TIPS: 保持各个请求会话 SESSION 独立不受干扰

2017年7月17日

Support browser contexts to launch different sessions #85

- <https://github.com/GoogleChrome/puppeteer/issues/85>
- <https://github.com/miyakogi/pyppeteer/issues/44>

23 天前

```
var context = await browser.createIncognitoBrowserContext();
var page = await context.newPage();

await page.goto('http://mail.aliyun.com');

var cookies = await page.cookies();
console.log(cookies)

await page.close();
await context.close();
```

隐身模式

```
browser.createIncognitoBrowserContext()
• to create new incognito context

browser.browserContexts()
• to get all existing contexts

browserContext.dispose()
• to dispose incognito context.
```

TIPS: 基于开发经验和MVC框架开发原则的去重技术

从URL中分离出 **静态/动态** 参数
动态参数就是攻击入口

“ AST | HASH ”

```
import ast

def var(x):
    try:
        if not isinstance(x, str):
            x = str(x)
        return ast.literal_eval(x)
    except:
        try:
            x = x.replace('\\', '\\\\')
            return ast.literal_eval("{}{}{}".format(x))
        except:
            return 'xz'
```

```
In [10]: type(var('123'))
Out[10]: int
```

```
In [11]: type(var('News'))
Out[11]: str
```

- 孤立数字出现的情况, 90%是动态参数
<https://tsrc.com/index.php/blog/msg/34>
- 数据类型相似, 长度一致 (md5 uuid hash)
</about.htm?profileId=f30dc1ad-4b53-4fb7-a1e0-adcd2c>
</about.htm?profileId=17ac5a75-9469-49a1-bf7d-52da38>
- 抽象语法树: 伪静态
<http://www.discuz.net/thread-3841114-1-1.html>
<http://www.discuz.net/{str:6}-{int:7}-{int:1}-{int:1}.html>
- HASH化去重做出出现频次统计, 反向排除
<http://xz.aliyun.com/show/id/123>
<http://xz.aliyun.com/show/id/456>
<http://xz.aliyun.com/show/id/789>
 - <http://xz.aliyun.com/{}/id/123>
 - <http://xz.aliyun.com/show/{}/123>
 - <http://xz.aliyun.com/show/id/{}>

Q&A

RE Q

正则表达式能解决的问题
就不要HACK底层了

