mitmproxy.org
How MITMproxy has been slaying SSL Dragons

Jim Cheetham
University of Otago
Information Security Office
jim.cheetham@otago.ac.nz

OWASP
April 14 2012

Copyright © The University of Otago
Permission is granted to copy, distribute and/or modify this
document under the terms of the Creative Commons
Attribution-ShareAlike 3.0 New Zealand (CC BY-SA 3.0) licence.

The OWASP Foundation  http://www.owasp.org
Introduction

- What is MITMproxy?
- Why is it useful?
- Dragon-slaying successes
- How does it work?
- How do we use it? (Demos)
What is MITMproxy?

- “An SSL-capable man-in-the-middle proxy”
- Generic pentest/debug tool
- Interactive, console based – intercept & modify
- Passive – like tcpdump/tshark
- Replay previous data
- Preserve cookies & authentication
- **Extensible** – invoke Python modules
  - Or system commands
- Programmable via libmproxy
Not just good looks

```bash
GET https://github.com/
  → 200 text/html 5.52kB
GET https://a248.e.akamai.net/assets.github.com/stylesheets/bundles/github2-24f59e3ded11f2a1c7ef9ee730882bd8d550cfb8.css
  → 200 text/css 28.27kB
GET https://a248.e.akamai.net/assets.github.com/images/modules/header/logov7@4x-hover.png?1324526958
  → 200 image/png 5.55kB
>> GET https://github.com/twitter

[7] [i:*] :help [*:8080]
```
Project maturity

- Initial: v0.2 – March 2010
- Current: v0.8 – 9 April 2012
- License: GPL v3 (+OpenSSL)
- Author: Aldo Cortesi
  - Network security, penetration testing, security architecture, source audits, risk assessment, software development
Why is MITMproxy useful?

- Balance of power has shifted
  - The browser is not the only user of HTTPS
  - The mobile platform is not generic

- HTTP-using code is increasingly opaque
  - Large JavaScript 'applications'
  - Application Stores: “Free = Ads”

- Existing tools are inflexible
  - Wireshark is great, but read-only
  - WebScarab ZAP, Fiddler, Paros, Charles – all GUI
  - Writing your own site-specific tests?
Feature highlights

- Rich filtering language
  - ~t “text/html”
  - ~hs Set-Cookie

- Anticache, anticomp
  - Makes sure you capture *all* the traffic

- Replay
  - Server-side replay fixes date, expired, last-modified

- Nice display modes
  - Hex, EXIF, JSON, JS
Dragon-slaying success

- Apple GameCenter
  - Super MegaWorm highscore
  - Game switched to OpenFeint

- Apple UDID
  - OpenFeint de-anonymized
  - https://api.openfeint.com/users/for_device.xml?udid=XXX
  - Apple withdrew UDID

- Apple Contacts data
  - path.com, Hipster
  - Apple withdrew implicit permission
How does MITMproxy do its job?

- Standard Proxy instance
  - A Proxy is a Man-in-the-Middle
  - But a proxy does not usually rewrite data
- HTTPS = TLS + HTTP
- MITM = TLS Man-in-the-Middle
- SSL Certificates are changed
  - But who checks the CA provenance?
HTTPS = TLS + HTTP

- Establish TLS first
- Then use normal HTTP in the TLS session
 TLS Man-in-the-Middle

- Default: just present a fake certificate
- Transparent: read data from the upstream certificate, and insert into the fake
How do we use MITMproxy?

- MITMproxy is **not** an attack tool
- Explicitly configure it as a proxy
  - Transparent proxy in v0.8
  - This is not ideal – data is lost
- Import the (auto-generated) CA Root cert
  - Or configure your own sub-CA
  - Or just keep pressing [Accept] – if you can
- Run as interactive console app
  - Or standalone 'mitmdump'
  - Or in your own Python code
The Android Problem

- Android provides no API for proxy settings
  - Bug 1273, 2008
- Android provides no root access
  - Rooting has implications for support/warranty
- Transparent proxy solutions are available
  - SNI is of course broken
  - Intent is lost: URLs → IP addresses
- Large sites rely heavily on SAN fields
- v0.8 introduces --upstream-cert
Demos

- Inspect a browser session
  - Login to a fastmail.fm account

- Intercept and edit a request or response
  - i ~u google.*/search
  - Edit 302 Location: response header
Demos

Run all traffic through a script

- nowasp.py

```python
import re

def request(context, flow):
    if (re.search('google\.\.', flow.request.host) and
        re.search('/search', flow.request.path)):
        flow.request.path = re.sub('q=OWASP','q=Anonymous',
                                    flow.request.path,0,re.IGNORECASE)

$ mitmdump -s nowasp.py
```
Demos

- **Client replay / hotspot wifi login**
  
  $ mitmdump -w session.mf
  
  $ mitmproxy -r session.mf
  
  - '#d' to delete un-needed requests
  - '#eb' to edit a request body
  - '#w' to save the final flow
  
  $ mitmdump -c session.mf

- **BUT other tools are still useful :-**

  $ echo "" | POST http://1.1.1.1/logout.cgi | grep 'You have used'
  
  You have used 1:37
Demos

- Upside-Down-Ternet
  - Identify data by Content-Type
    - Images (but not CSS sprites)
      - Pass through OS utility ('convert' from ImageMagick)
      - Or use native Python modules (as per project docs)
    - Text (according to BeautifulSoup)
      - Pass through upsidedown.py
      - Decode the page first!
import re
import subprocess
from BeautifulSoup import BeautifulSoup
import upsidedown

def response(ctx, flow):
    try:
        # Flip images using ImageMagick
        if re.match('image/',flow.response.headers["content-type"][0]):
            proc = subprocess.Popen('/usr/bin/convert -flip - -',
                                shell=True,
                                stdin=subprocess.PIPE,
                                stdout=subprocess.PIPE,)
            flow.response.content=proc.communicate(flow.response.content)[0]
            proc.stdin.close()

        # Flip text using BeautifulSoup and upsidedown
        if re.match('text/html',flow.response.headers["content-type"][0]):
            flow.response.decode()
            soup = BeautifulSoup(flow.response.content)
            for text in soup.findAll(text=True):
                text.replaceWith(upsidedown.transform(text))
            flow.response.content=str(soup)

    except IndexError:
        ctx.log("no content-type[0]")
Resources

- http://mitmproxy.org/
- https://github.com/cortesi/mitmproxy
- http://corte.si/