uamd Documentation

Release 0.1

2011, Alisue

December 01, 2011

CONTENTS

uamd is a library for detecting device via HTTP_USER_AGENT. Currently this library can detect the device listed below

- DoCoMo mobile phone
- KDDI mobile phone
- SoftBank mobile phone
- iPhone/iPod Touch/iPad
- Android
- Internet Explorer
- Google Chrome
- Firefox
- Opera
- Lunascape

The device doesn't listed above will treat as `device.DummyDevice` which will treated as device which support cookie and has no carrier (treated like as PC Browser)

CHAPTER

ONE

GETTING STARTED

If you are new to uamd, you may want to start with these documents to get you up and running

1.1 Getting started with uamd

Detecting the device accessed via HTTP_USER_AGENT is required to build web site for mobile phone (so you can change template, css, javascript or whatever).

However even detecting the device is very important function, the mechanisms of it is not fun all :-p

That's why I create reusable detecting library. This library is mainly created for django-mfw so please check out it as well.

This tutorial assumes that you are using Django but even if you are not familia with it you may able to understand what's going on I think.

1.1.1 Basic usage

If you have HTTP Headers dictionary in `request.META`, you can detect the device with the code below:

```
>>> print request.META
{'HTTP_USER_AGENT': u"DoCoMo/1.0/F504i/c30/TD", ...}
>>> device = uamd.detect(request.META)
>>> print device
DoCoMo F504i
```

1.1.2 Advanced usage

Using device information

the device `uamd.detect()` return has properties listed below.

support_cookie if the device can handle cookie. this value doesn't reflect actual state so even if user turned off cookie, it may return `True` for the device which usually support cookie.

carrier the carrier name detected. supported carrier is 'docomo', 'kddi' and 'softbank'. the device which doesn't have carrier will return `None`

uid the carrier's UID for the connection. make sure the `spoof` is not set to `True` see *What is the spoof* for more detail

spoof return `True` if the device may spoofed. see *What is the spoof* for more detail

version if the device has version.

model if the device has model

Caution: for convinience, device instance doesn't raise AttributeError even if unknown attribute is called. Insted of AttributeError, device will return `None`. That's why the code below doesn't work as expect:

```
# Doesn't work !!!
if hasattr(device, 'foobar'):
    print "Worked"
# Work correctly
if device.foobar is not None:
    print "Worked"
```

Using custom CIDR loader

Default CIDR loader fetch CIDR data from each carrier's web site and stored the data in the instance as cache. However the way of doing is not efficient when the instance is released a lot (like Django)

So I recommended to create your own CIDR loader for your application. The code below is Databased CIDR loader for Django from django-mfw and the full source code is found on https://github.com/alisue/django-mfw/blob/master/mfw/core/cidr.py:

```
class DatabaseLoader(uamd.cidr.Loader):
    u"""Loading cidr via carrier from Django's Database and Internet"""
    _EXPIRE = relativedelta(weeks=1)
    _DELIMITER = u"\n"

    def _str_to_data(self, s):
        values = s.split(self._DELIMITER)
        return map(lambda x: IP(x), values)
    def _data_to_str(self, d):
        values = map(lambda x: unicode(x), d)
        return self._DELIMITER.join(values)

    def get(self, carrier):
        expire = datetime.now() - self._EXPIRE
        cache = getattr(self, '_data_cache', None)
        if cache and cache[0] > expire:
            return cache[1]
        try:
            cidr = CIDRCache.objects.get(updated_at__gt=expire, carrier=carrier)
        except CIDRCache.DoesNotExist:
            cidr = CIDRCache.objects.get_or_create(carrier=carrier)[0]
            cidr.data = self._data_to_str(self.fetch(carrier))
            cidr.save()
        data = self._str_to_data(cidr.data)
        updated_at = datetime.now()
        setattr(self, '_data_cache', (updated_at, data))
        return data
```

What is the spoof

A lot of web framework use cookie for storing session_id or whatever. Using cookie is the best way in security aspect. But some of device cannot handle cookie and they cannot store session_id. That's why some web framework or web site use carrier's UID (User ID) for storing session_id.

However UID commonly send from carrier server as HTTP X Header and you know HTTP X Headers can modify with Firefox plugin or whatever if you are accessing the web site with Computer.

So most of web framework or web site check IP Address and make sure the IP Address is in CIDR of mobile phone to ignore access to mobile web site from Computer.

Is it secure? Recently some mobile device has thethering function and if you connect the site via thethering. Who know that you are accessing with Computer.

To reduce this security risk, uamd check IP Address and make sure the IP Address is in **carrier's CIDR** insted. What the difference?

Checking carrier's CIDR or CIDR of all mobile carrier makes big difference. Most carrier server **rewrite** HTTP X Headers for handling own carrier's UID. So if you are accessing with in DoCoMo's CIDR, the HTTP_X_DCMGUID(DoCoMo's HTTP Headers for handling UID) header is rewrited by carrier and user never be able to fake it. But if you are accessing with in KDDI's CIDR, the carrier doesn't know about HTTP_X_DCMGUID so the value can be faked by user.

That's why uamd set `device.spoof = True` when the device carrier detected via HTTP_USER_AGENT doesn't connected from in CIDR of the carrier detected. So check this value before you use `device.uid` for saving session_id or whatever.

1.2 Device Reference

1.2.1 Builtin Device

The buildin device is listed below.

`uamd.device.mobile`

CHAPTER

TWO

TABLE OF CONTENTS

2.1 uamd API Reference

Device detection library via HTTP_USER_AGENT

2.1.1 Introduction

uamd is library for detecting device via HTTP_USER_AGENT.

Currently this library can detect the device listed below

- DoCoMo mobile phone
- KDDI mobile phone
- SoftBank mobile phone
- iPhone/iPod Touch/iPad
- Android
- Internet Explorer
- Google Chrome
- Firefox
- Opera
- Lunascape

The device doesn't listed above will treat as `device.DummyDevice` which will treated as device which support cookie and has no carrier (treated like as PC Browser)

2.1.2 Methods

CHAPTER

THREE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*