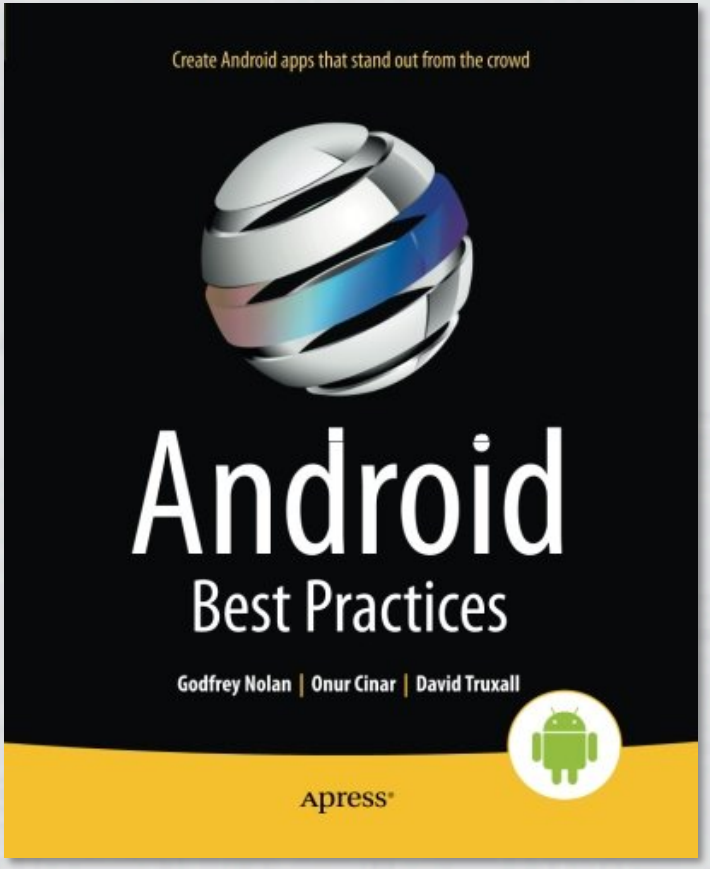
Mobile Authentication in the Web World

David Truxall, Ph.D.

bit.ly/mobileauth

About Me



@davetrux

blog.davidtruxall.com

You

May be familiar with Android and/or iOS

Might be a web developer

Probably have web sites at work

Want to use an existing user identities

Not an Identity/Security guru

Why?

Enabled informed choices

Understand web authentication

Not re-inventing the wheel

Existing technologies make this easy

Agenda

1. Basic security
2. HTTP basics
3. Types of web-based authentication
4. Mobile Consumption

Code examples are contrived to show the flow and basic technique

OWASP Mobile Top 10

1. Weak server side controls
2. Insecure Data Storage
3. Lack of Transport Security
4. Unintended Data Leakage
5. Poor Authentication and Authorization

OWASP Mobile Top 10

6. Broken Cryptography
7. Client Side Injection
8. Security Decisions via Untrusted Input
9. Improper Session Handling
10. Lack of Binary Protections

Security

Don’t

- Store the password
- Send the password
- Own the password
- Invent your own

Security

Do

- Use transport security (SSL)
- Implement sessions
- Store on the server not client

HTTP Anatomy

Request

- Method
- URL
- Querystring

Headers

- Cookies, Authorization

Body

- HTML, JSON, XML, Multi-part Form

HTTP Request Methods

GET

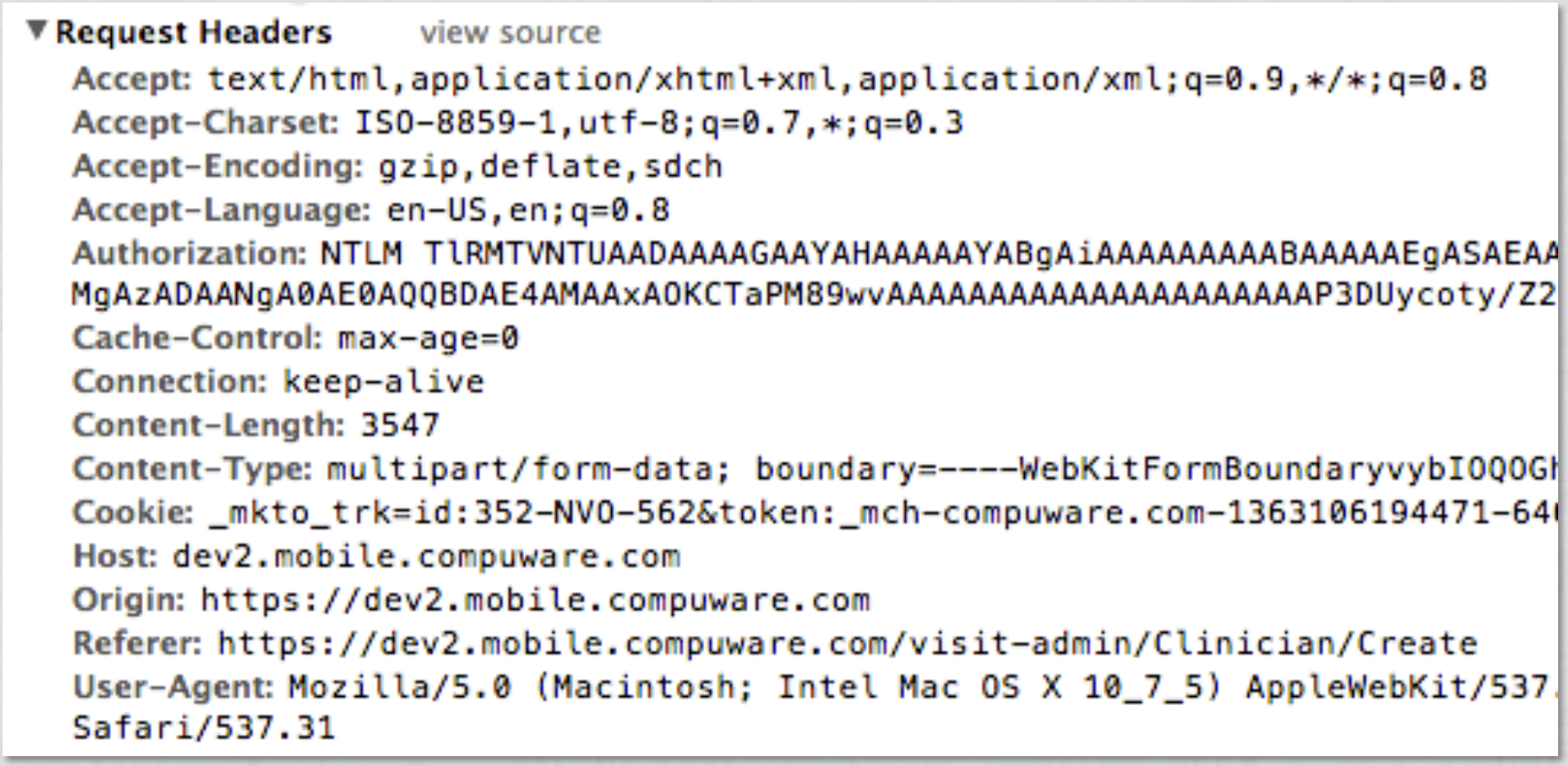
POST

PUT

DELETE

OPTIONS

HTTP Headers



HTTP Body

▼ **Request Payload**

```
------WebKitFormBoundaryvybIOQOGhl0ceumG
Content-Disposition: form-data; name="Id"

0
------WebKitFormBoundaryvybIOQOGhl0ceumG
Content-Disposition: form-data; name="photoUpload"; filename="Red Apple.gif"
Content-Type: image/gif


------WebKitFormBoundaryvybIOQOGhl0ceumG
Content-Disposition: form-data; name="FirstName"

Test
------WebKitFormBoundaryvybIOQOGhl0ceumG
Content-Disposition: form-data; name="LastName"

Data
------WebKitFormBoundaryvybIOQOGhl0ceumG
Content-Disposition: form-data; name="Description"

Sit aliquet? Turpis odio amet ultricies porttitor velit sagittis in, ut phase
itor integer et phasellus odio nec auctor! Ultricies in, amet proin et, cras.
assa? Sociis risus nunc, pellentesque tristique dignissim cras. Auctor, portt
us augue augue, placerat integer pulvinar nunc vel porta tortor ultrices sagi
am? Rhoncus adipiscing pulvinar? Adipiscing scelerisque a nunc augue odio! Nu
sed lundium mid, lorem! Magnis turpis ac auctor hac ridiculus, cursus, magna,
------WebKitFormBoundaryvybIOQOGhl0ceumG--
```

Response Codes

200	OK	We're good
302	Redirect	Over there
400	Bad Request	Your fault
401	Unauthorized	Not for you
404	Not found	Not here
500	Error	My fault

Authentication Methods

Forms

Basic

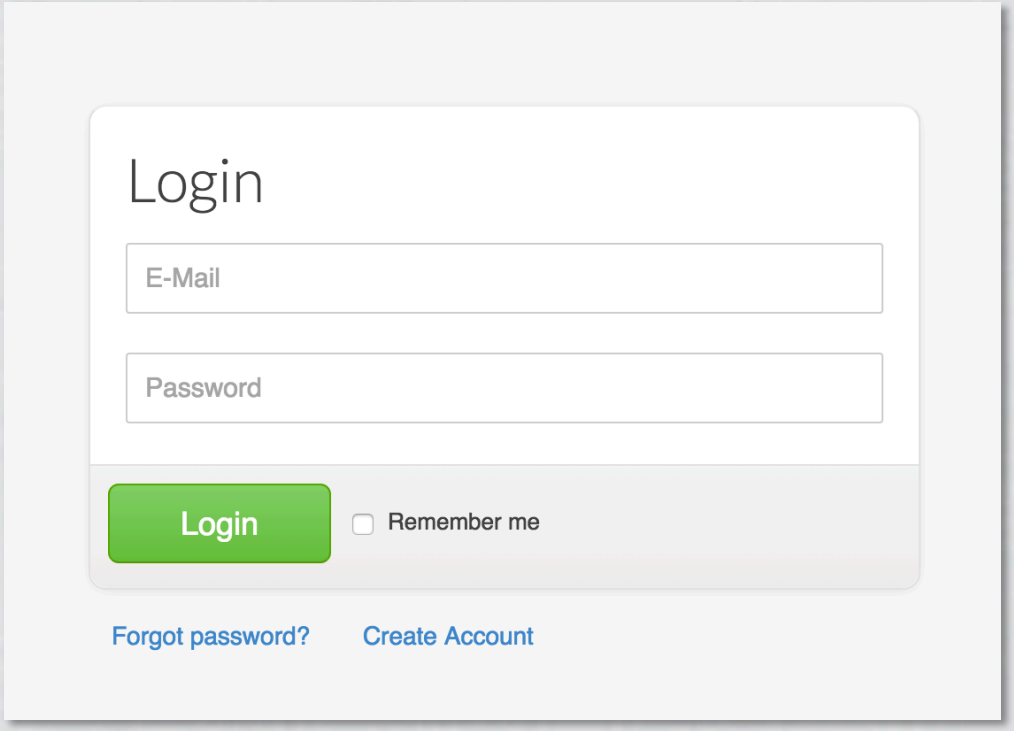
Digest

NTML

OAuth

HMAC

Forms Authentication



HTTP Basic

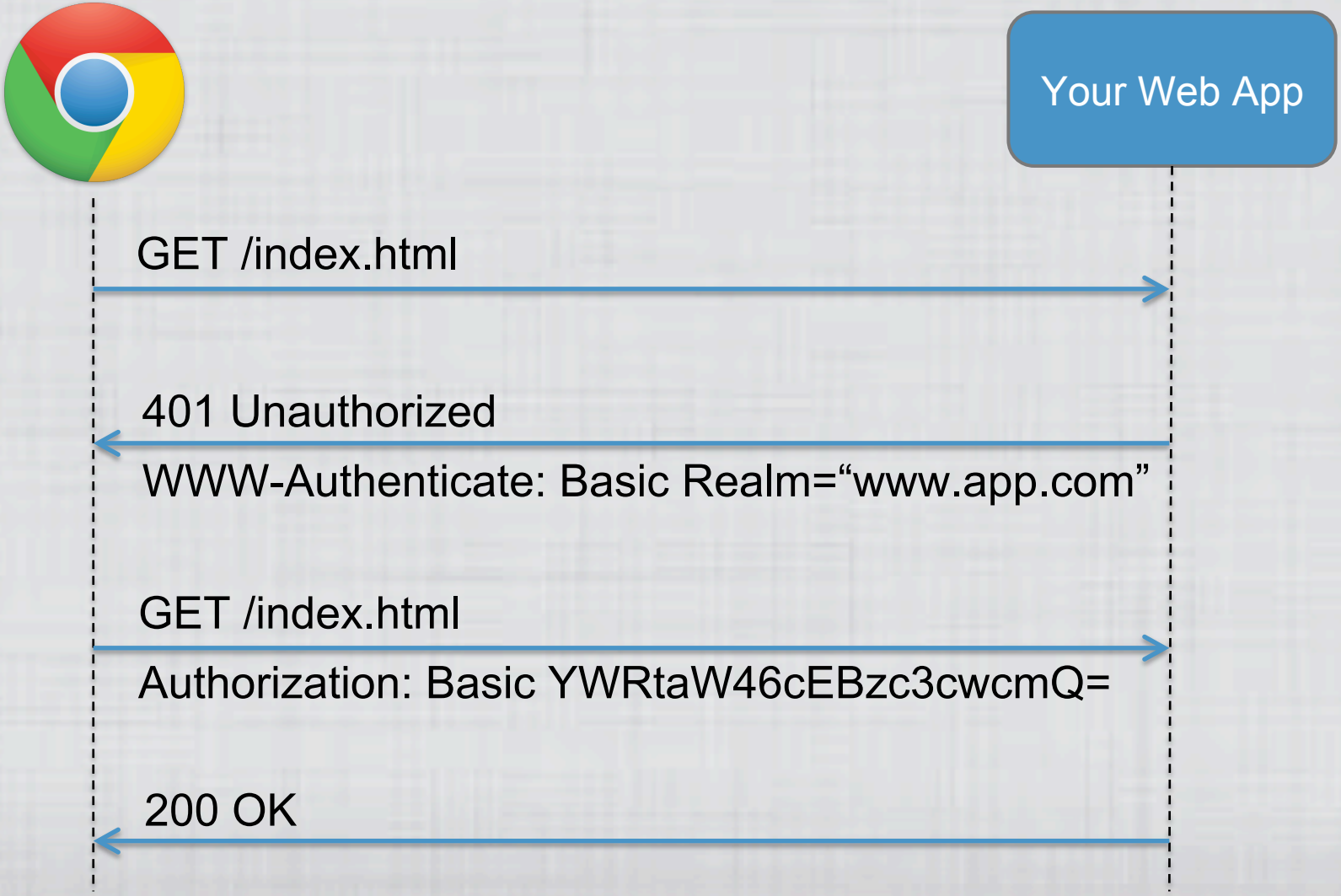
Relies on SSL

Clear text

Sends the password

Both iOS and Android handle this in APIs

HTTP Basic


Your Web App
GET /index.html
401 Unauthorized
WWW-Authenticate: Basic Realm="www.app.com"
GET /index.html
Authorization: Basic YWRtaW46cEBzc3cwcmQ=
200 OK

HTTP Basic

1. Concatenate username and password
2. Encode them in Base64
3. Prefix this string with ‘Basic’
4. Add as Authorization HTTP header

Authorization: Basic YWRtaW46cEBzc3cwcmQ=

Today's Web Service


```
[{
    “gender":"m",
    "firstName":"Ron",
    "lastName":"Lynn”
    },
    {
    "gender":"f",
    "firstName":"Pauline",
    "lastName":"Schultz”
    },
    {
    "gender":"f",
    "firstName":"Muriel",
    "lastName":"Hooper”
    }
]
```

HTTP Digest

Stronger than Basic

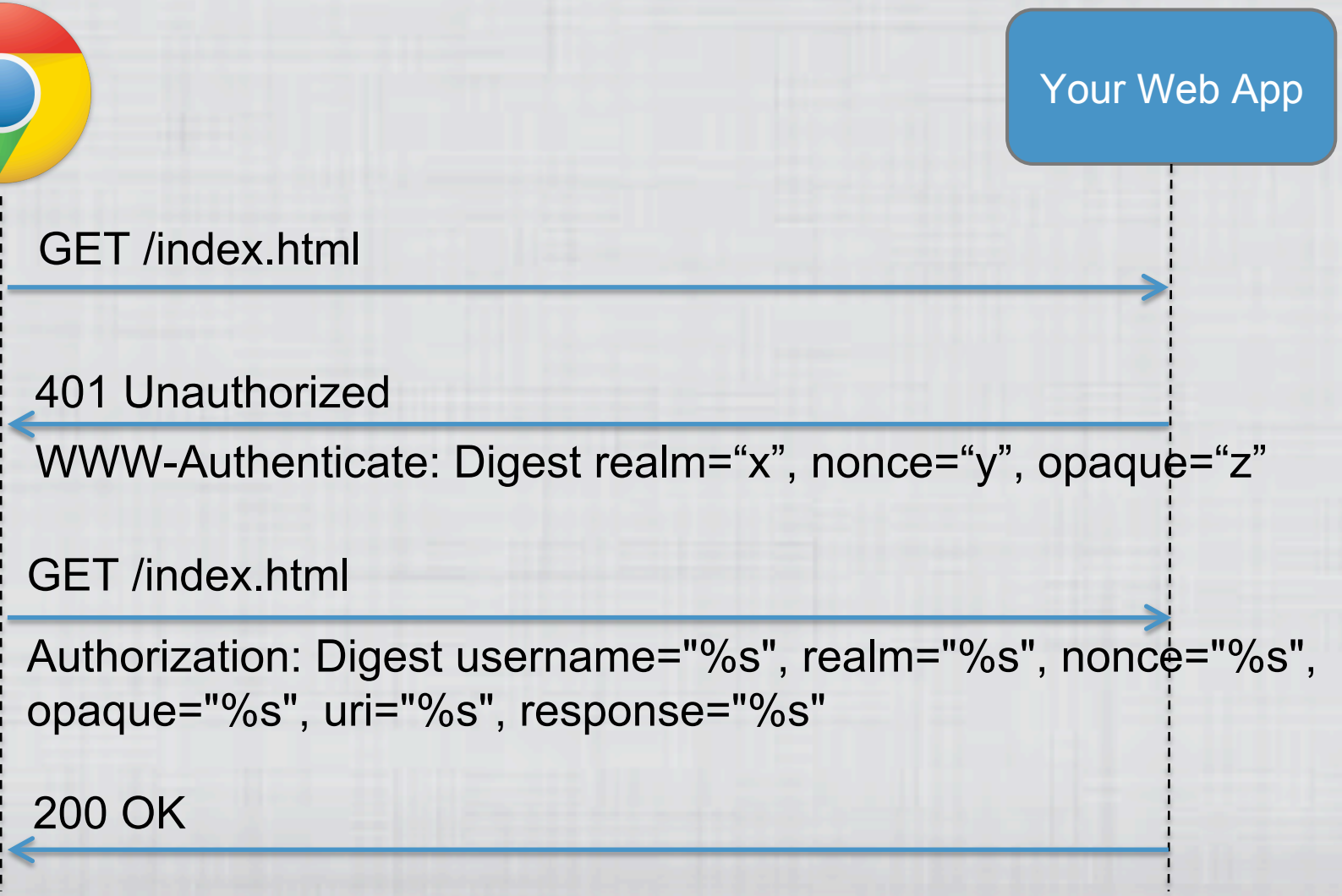
No requirement for SSL

Password not sent

Uses MD5 hashing

Enhancements optional

HTTP Digest


Your Web App
GET /index.html
401 Unauthorized
WWW-Authenticate: Digest realm="x", nonce="y", opaque="z"
GET /index.html
Authorization: Digest username="%s", realm="%s", nonce="%s", opaque="%s", uri="%s", response="%s"
200 OK

HTTP Digest

Server sends nonce, opaque and realm

A1 = MD5(“username:realm:password”)
A2 = MD5(“method:uri”)
response = MD5(A1:nonce:A2)

Authorization: Digest username="%s", realm="%s", nonce="%s", opaque="%s", uri="%s", response="%s"

NTLM Authentication

Microsoft-based

Active Directory

No requirement for SSL

Password not sent

Better than Digest

Complicated calculation

NTLM in Android

Apache client, not HttpUrlConnection

JCIFs library

NTLM in iOS

No library needed

Implement NSURLSessionDelegate

OAuth v2

Open standard for authorization

Delegation of authorization

Third parties use an authorization source

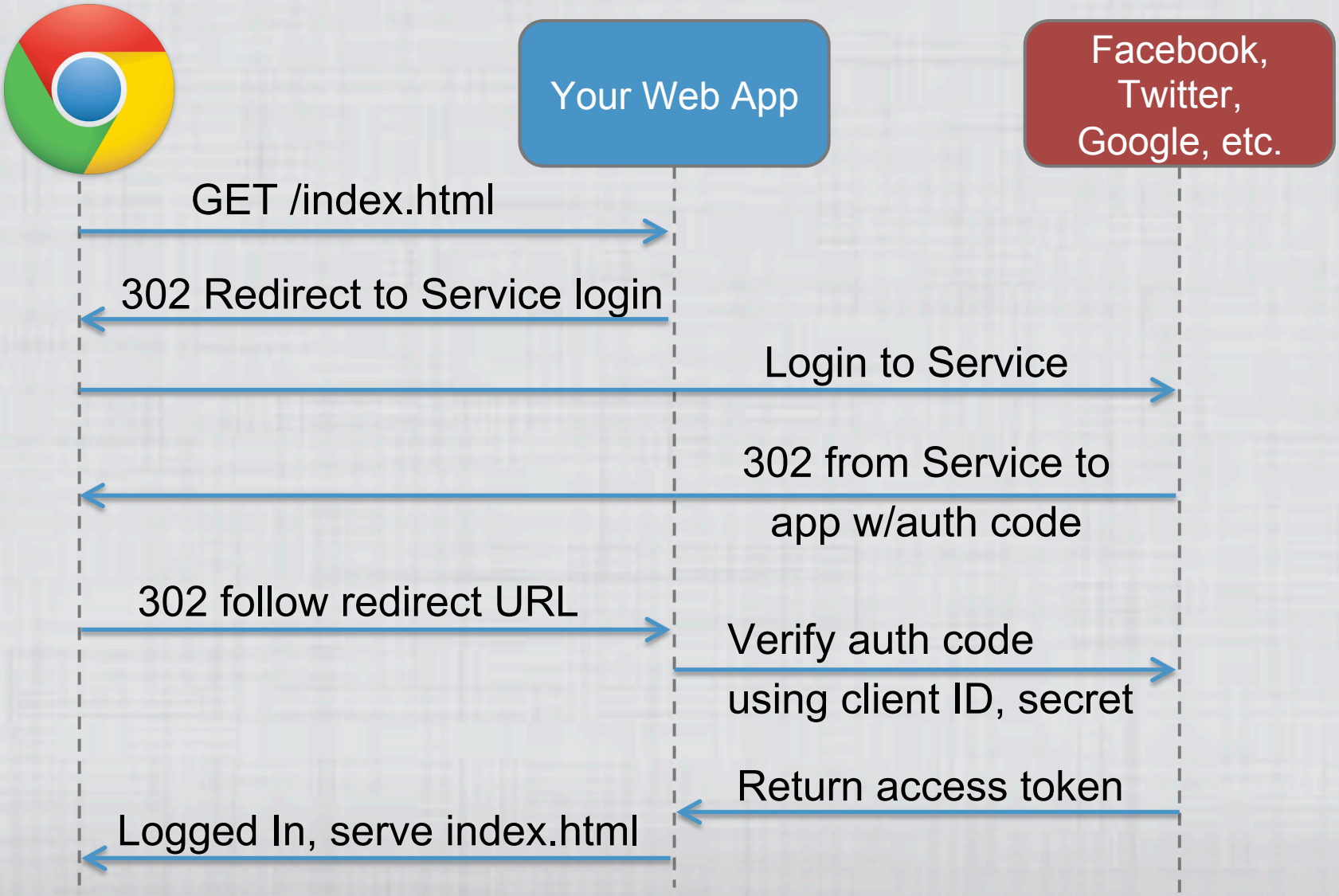
Invented for web sites

Password sent once

OAuth

the code is more what you'd call "guidelines" than actual rules

Typical OAuth Web Flow



Simple OAuth Web Service Flow

Your Web Services

OAuth Provider

Login to Provider

New access & refresh tokens

GET /order/1234

Authorization: Bearer d23a7726-36…

Verify token with client ID, secret

200 OK

{ id: 1234, quantity: 7, …}

Send refresh token

New access & refresh tokens

HMAC

Hash-based Message Authentication Code

Guarantee authenticity of message

A shared secret key – both client and server

No need for SSL

AWS

Password not sent

Authorization: HMAC trux:44CF006BF252F707:jZND/A/f3hSvVzXZjM2HU=

HMAC on Client

Create a request

POST /customer/ { id: 123, orders: 6, …}

Create a signature using shared key

base64(hmac-sha1(verb + headers + content)

Add as authorization header

Authorization: HMAC userName:signature

HMAC on Server

Retrieve key from DB based on userName

Recreate signature based on request
base64(hmac-sha1(verb+ headers + content)

Compare signatures

Derived Credentials

Replacement for actual devices/badges

An alternative token implemented and deployed directly on mobile devices

Security Assertion Markup Language

SAML

XML-based

Not just HTTP

Shibboleth, ADFS

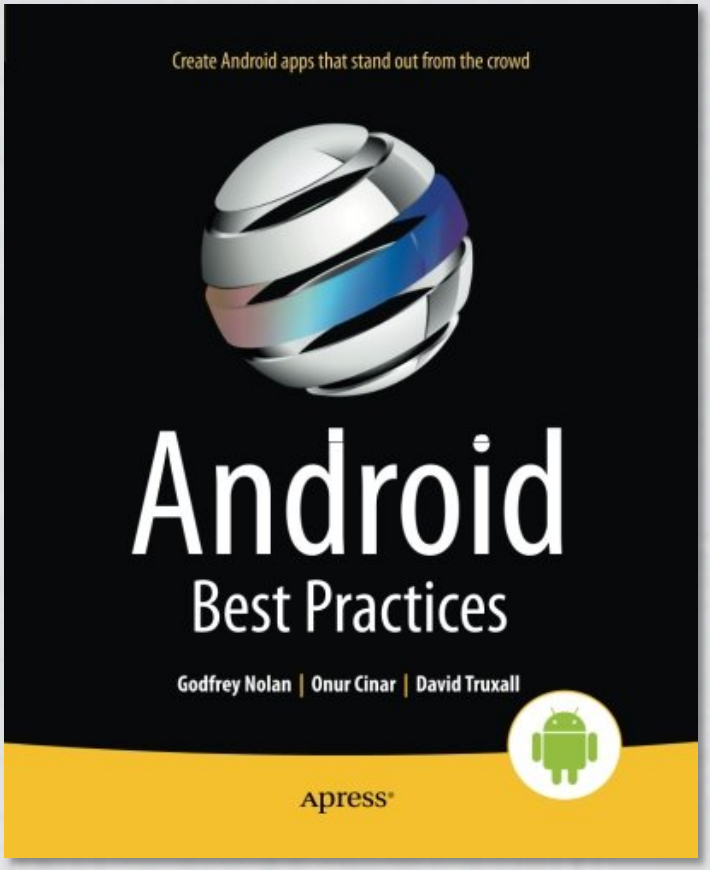
Defined in 2005

Also intended for web applications

Call It

Which one should you use?

Yet Another Shameless Plug



@davetrux

blog.davidtruxall.com

bit.ly/mobileauth

Resources

HTTP Digest

NTLM

SAML vs OAuth

Derived Credentials

RESTful Cookbook

Android HTTP Client