Application Firewalling in the Age of Mobile: New Considerations

- OWASP Orlando Chapter Meeting
  May 15, 2012
  Stephen Mak, Product Manager
  smak@layer7tech.com

Agenda

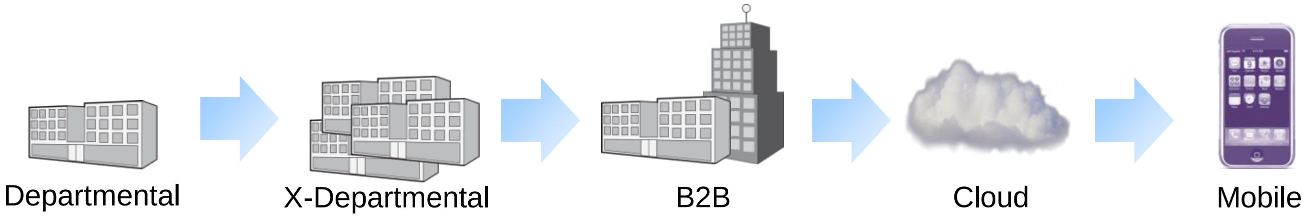
- Introduction
  - Layer 7 Technologies
- Background
  - Hacking's Gilded Age: How APIs Will Increase Risk and Chaos
- The New API Threat
  - API Parameterization
  - Identity
  - Cryptography

Layer 7 Technologies – Introduction

- Founded in 2002
- Based in Vancouver, BC, Canada
- Provide Integration Governance solutions for SOA, Cloud and API Management
- Customers include large commercial enterprises, public sector organizations, and service providers across a number of vertical markets worldwide

Layer 7 Technologies Provides Secure Integration Technology That Enable The Hybrid Enterprise: Divisions, Partners, Mobile & Cloud

Evolution of Enterprise Connectivity:

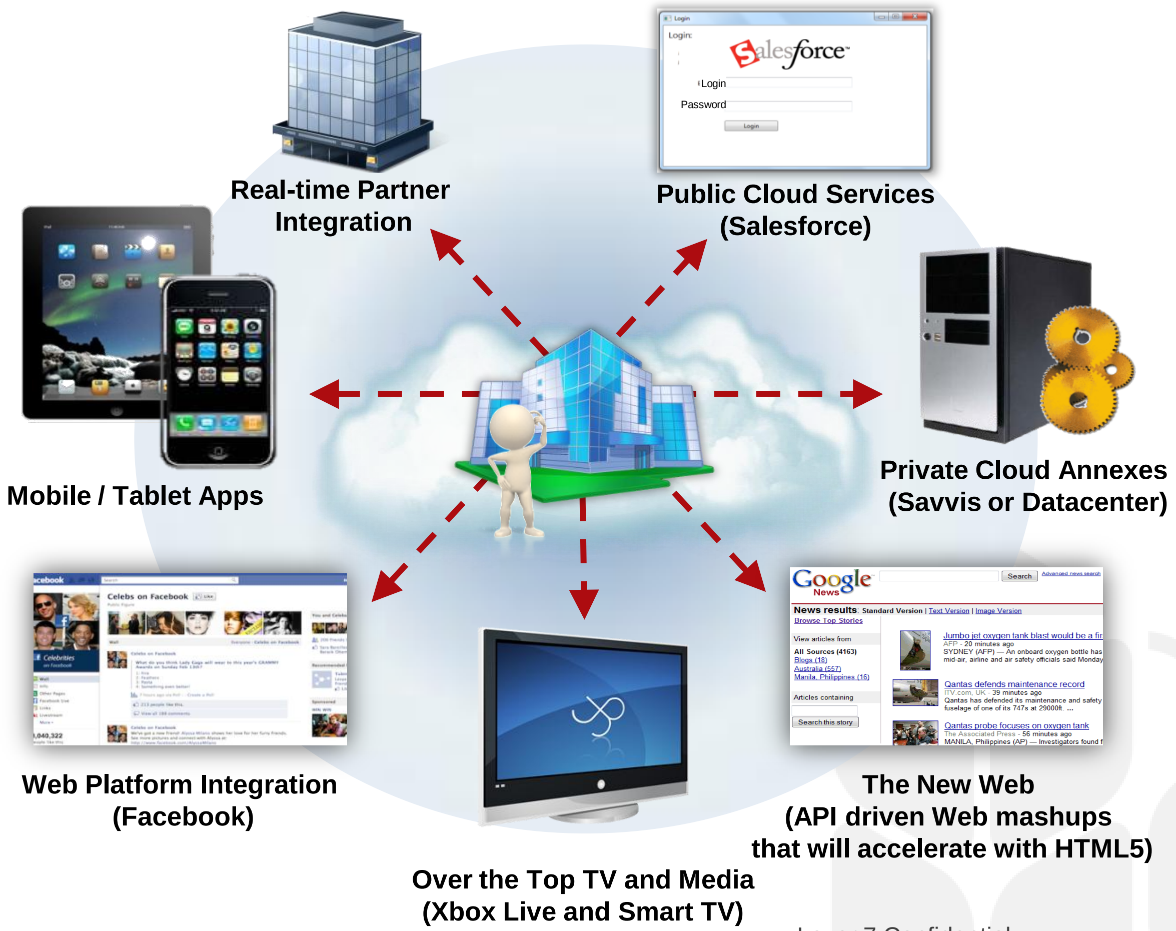


Hybrid Challenges :

- Protection of applications exposed over internet
- Reuse of information shared across departments, partners, mobile & Cloud
- Ease of integration: reconciling disparate identity, data types, standards, services
- Federated & Delegated Security
- Performance

Creating the demand for:

- Security
- Manageability of application APIs & interfaces
- Flexibility / ease of integration
- Governance of enterprise
- Cloud interactions



Real-time Partner Integration
Login
Login
Password
Public Cloud Services (Salesforce)
Mobile / Tablet Apps
Private Cloud Annexes (Savvis or Datacenter)
Celebs on Facebook
Web Platform Integration (Facebook)
Google News
Search
The New Web (API driven Web mashups that will accelerate with HTML5)
Over the Top TV and Media (Xbox Live and Smart TV)

Emerging Hybrid Enterprise Connectivity Use Cases

X-Departments & Agencies

- Application Reuse
- Private Cloud Integration
- Data Federation
- Federated ESB

Partners

- Real-time Supply Chain
- Media Syndication
- Trading Platform Integration
- Web Ecosystems

Mobile

- BYOD Employee Enablement
- iPad Field Enablement
- API Developer Communities
- Smart Grid

Cloud

- SaaS Access
- IaaS Integration & Governance
- Big Data Connectors
- Social Integration

L7 Product Suite Designed For New API-Driven Cloud and Mobile Connectivity Requirements

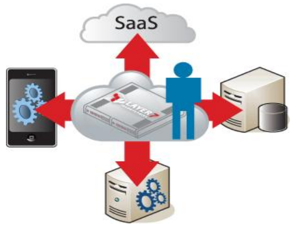
Virtual & Cloud Deployable API Gateway

x86 Hardware
(HP, Cisco, Dell, Sun…)

Software

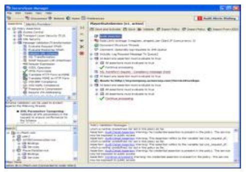
VMware vApp

Amazon Machine Image



Service

Simple & Flexible Security, Policy, Extensibility


Policy Manager

Identity Token Service / OAuth Toolkit

SDK / APIs

API Management & Governance

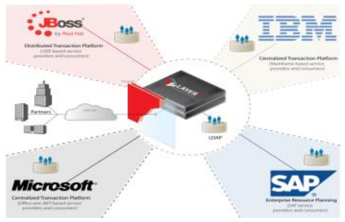
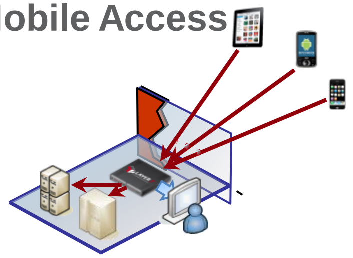
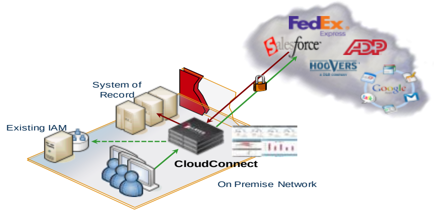
Enterprise Service Manager

API Developer Portal

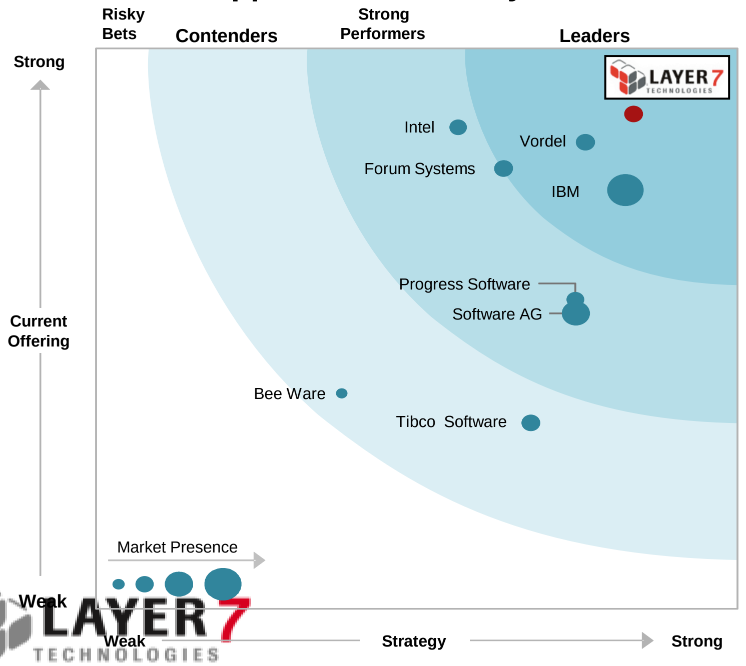
Integrated HSM Card

Layer 7 Is The Market Leader For Securing And Governing SOA, API And Cloud Based Interactions Across The New Hybrid Enterprise

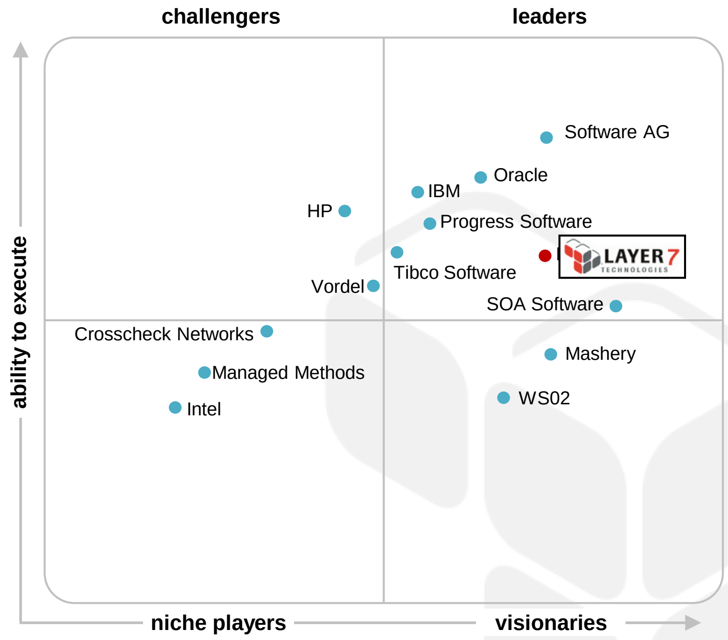
Problems We Address

X-Department Access	Partner Access	Mobile Access	Cloud Access
			
SOA Governance	SOA Governance & Security	API Management	Cloud Integration

The Forrester Wave Leader for SOA & API Application Gateways, Nov 2011



Gartner Magic Quadrant Leader For SOA Governance & API Management Technologies, Oct 2011



Examples of Layer 7 Technologies Customers

Financial Services	Communications	Public Sector	Select Others
MasterCard	swisscom	U.S. ARMY	BEST BUY
Baloise Group	bezeq	United States Coast Guard 1790	3M
Erie Insurance	BT	U.S. AIR FORCE	Adobe
SECURIAN	TÜRKSAT communication is everything	Department of Transportation United States of America	Alaska Airlines
AXA	Telstra	National Security Agency United States of America	CISCO
TransUnion	france telecom	U.S. Department of Homeland Security	Duke Energy
USAA	ROGERS	Federal Aviation Administration	carlson marketing
CHRYSLER FINANCIAL	TIME WARNER CABLE	Department of Justice Federal Bureau of Investigation	Reed Elsevier
unum	terremark	Department of Health & Human Services USA	Amerigroup RealSolutions in healthcare
PAYCHEX	SAVVIS A CenturyLink Company	Department of State United States of America	Lilly
De Nederlandsche Bank Eurosysteem		Office of the Director of National Intelligence United States of America	NVIDIA
EVEREST		NATO	GM
Northern Trust			UTi
Allstate You're in good hands.			aeroplan
Shell Trading			NORFOLK SOUTHERN
The First American Corporation			Sempra Energy
National Bank of Belgium			
RBS The Royal Bank of Scotland Group			
FannieMae			

LAYER 7 TECHNOLOGIES

Background / Context

- The Gilded Age?
- A time of impressive economic growth (output, 1865-1898):
  - Wheat: 256%
  - Corn: +222%
  - Coal: +800%
  - Railway: +567% (miles of track)
- Rise of the great “robber-barons”
  - Rockefeller
  - Mellon
  - Carnegie
  - Morgan
  - Vanderbilt
  - Astor
  - ...

Hacking’s Gilded Age: How APIs Will Increase Risk And Chaos

K. Scott Morrison
CTO & Chief Architect
Layer 7 Technologies

Session IDASEC-402
Session Classification: Intermediate

RSA CONFERENCE 2012

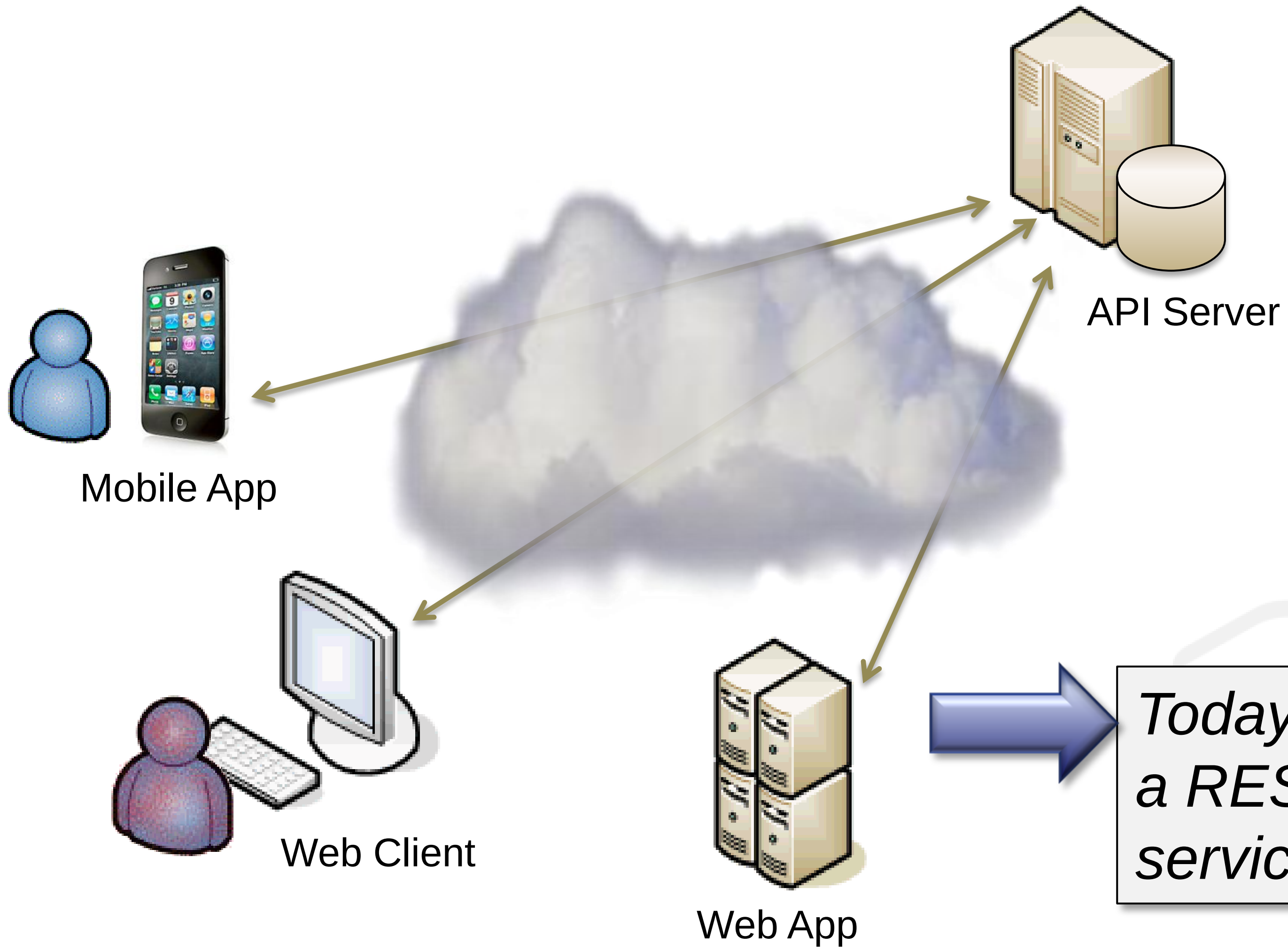
➔Who will be the robber-barons of the 21st century?
➔APIs could be a rich resource for exploitation

Source: http://en.wikipedia.org/wiki/Gilded_Age

Here Is What This Talk Is About:

- The new API threat
  - …and the potential rise of the hacker-robber-baron
- Are APIs just like the Web? Or are they different?
  - Look at three important areas:
    1. Parameterization
    2. Identity
    3. Cryptography
- How to apply the lessons of this talk

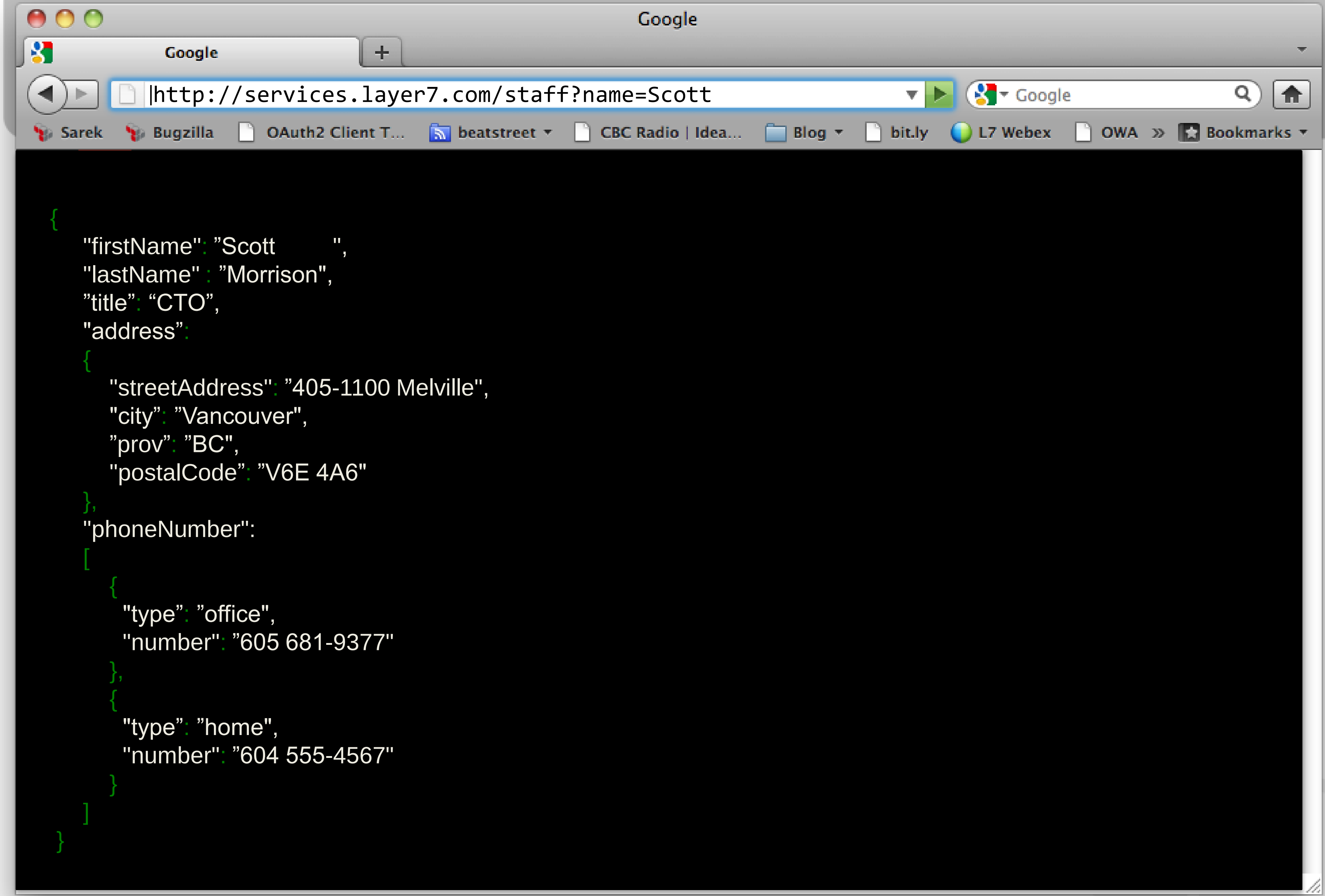
What is an API?



Today, an API is a RESTful service

For Example:

```
GET http://services.layer7.com/staff?name=Scott
```



```
{
    "firstName": "Scott        ",
    "lastName" : "Morrison",
    "title": "CTO",
    "address":
    {
       "streetAddress": "405-1100 Melville",
       "city": "Vancouver",
       "prov": "BC",
       "postalCode": "V6E 4A6"
    },
    "phoneNumber":
    [
       {
         "type": "office",
         "number": "605 681-9377"
       },
       {
         "type": "home",
         "number": "604 555-4567"
       }
    ]
 }
```

This is a Technological Sea Change

	Old	New
Transport	HTTP	HTTP
Data	XML	JSON
Authentication	Basic, X.509, Kerberos, SAML	OAuth
Confidentiality & Integrity	WS-Security	SSL

LAYER 7 TECHNOLOGIES

Where Simple Won

SOAP + WS-*

- Complex
- Highly standardized
- Vendor-driven
- Barriers

RESTful APIs

- Simple
- Informal
- Grassroots
- Frictionless

“Sounds great. So what’s the problem?”

The Real Problem Is This:

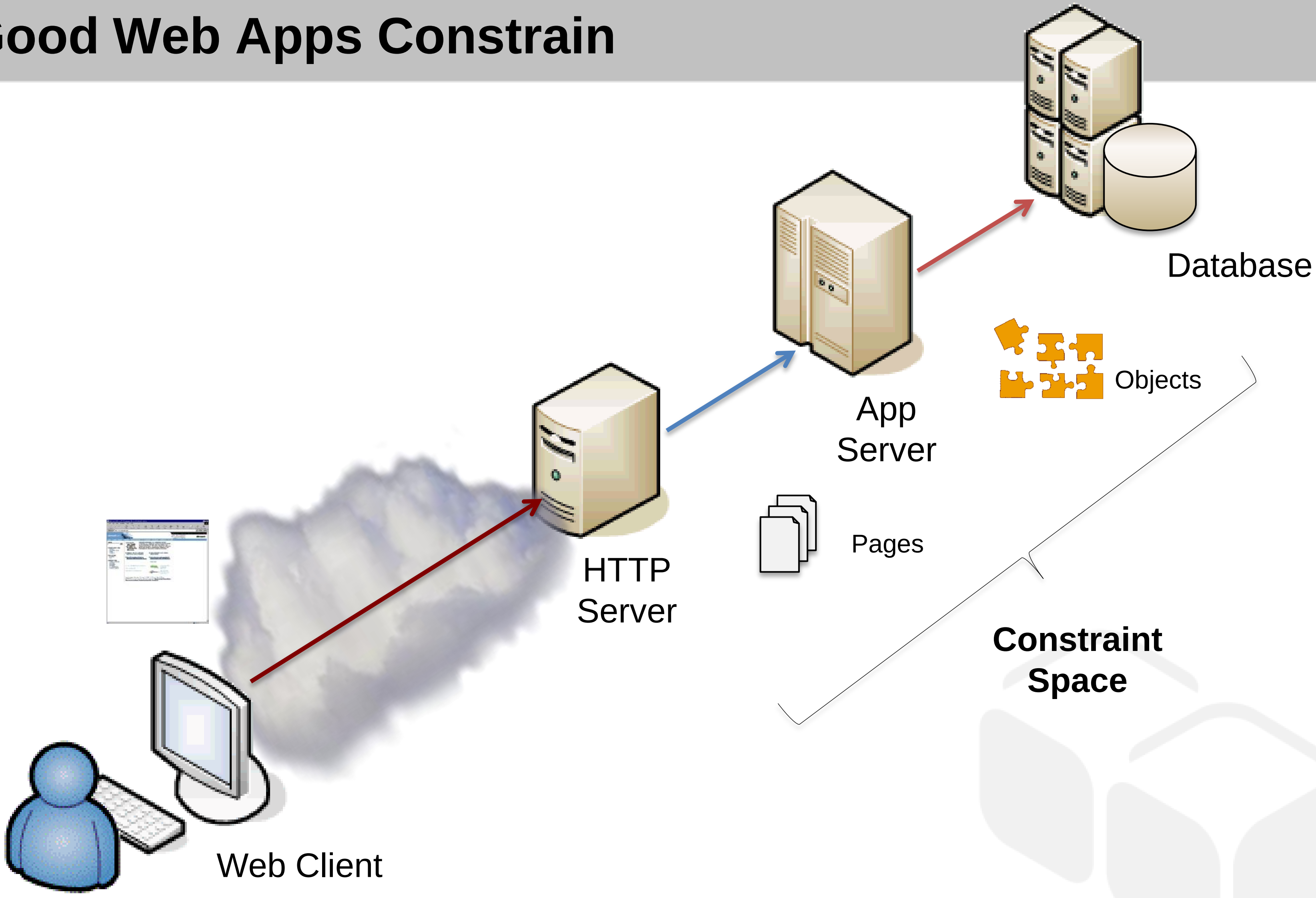
API Development !=
Web Development

In Particular:
We need to be wary of bad web development practices migrating to APIs…

Problem Area #1: *API Parameterization*

- In the web world, parameterization was limited and indirect
  - Subject to the capabilities of URLs and forms
- APIs in contrast and offer much more explicit parameterization
  - The full power of RESTful design: GET, POST, PUT, DELETE
    - (And don't stop there… what about effects of HEAD, etc)?
- This is a greater potential attack surface
  - Injection, bounds, correlation, and so on

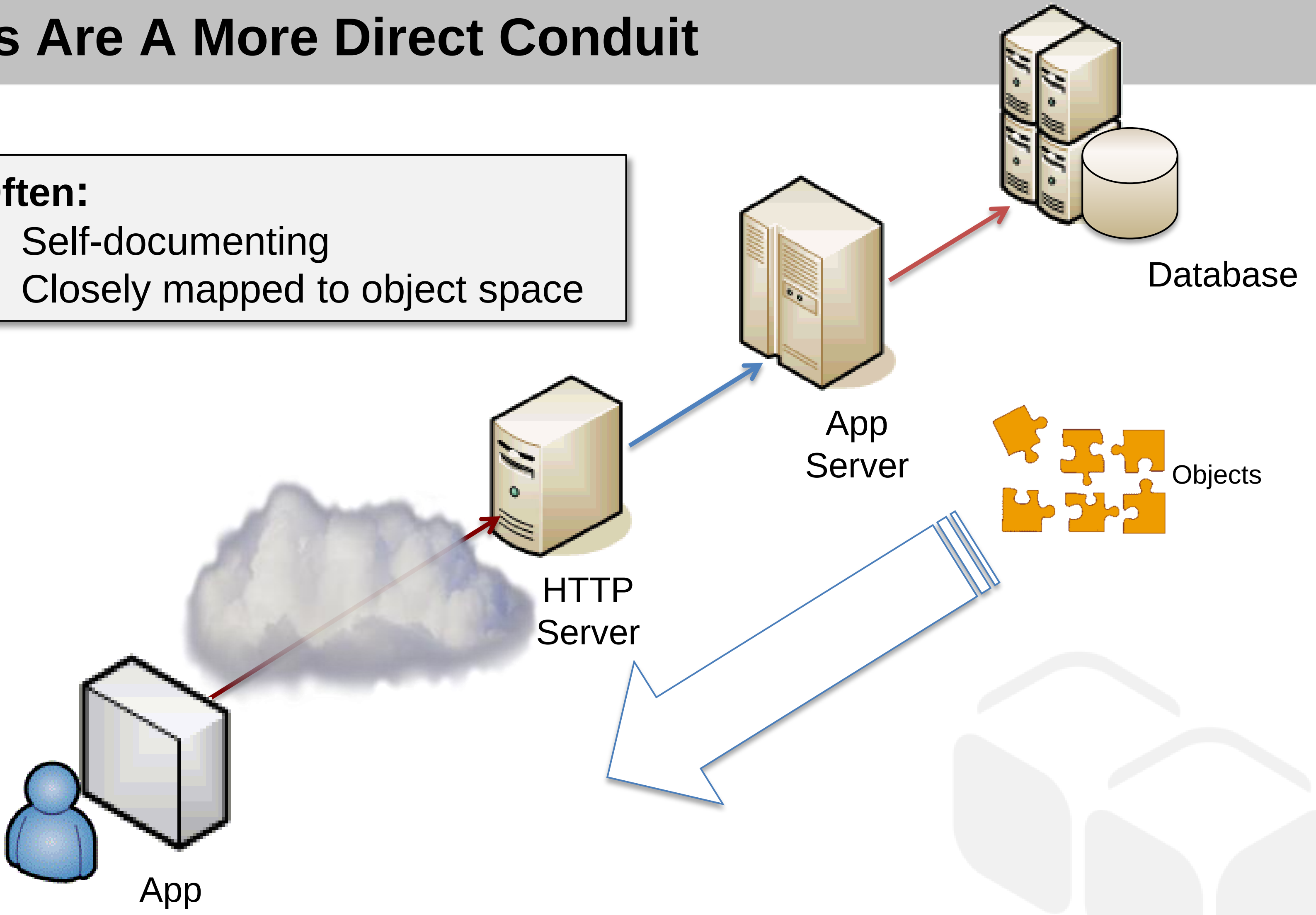
Good Web Apps Constrain



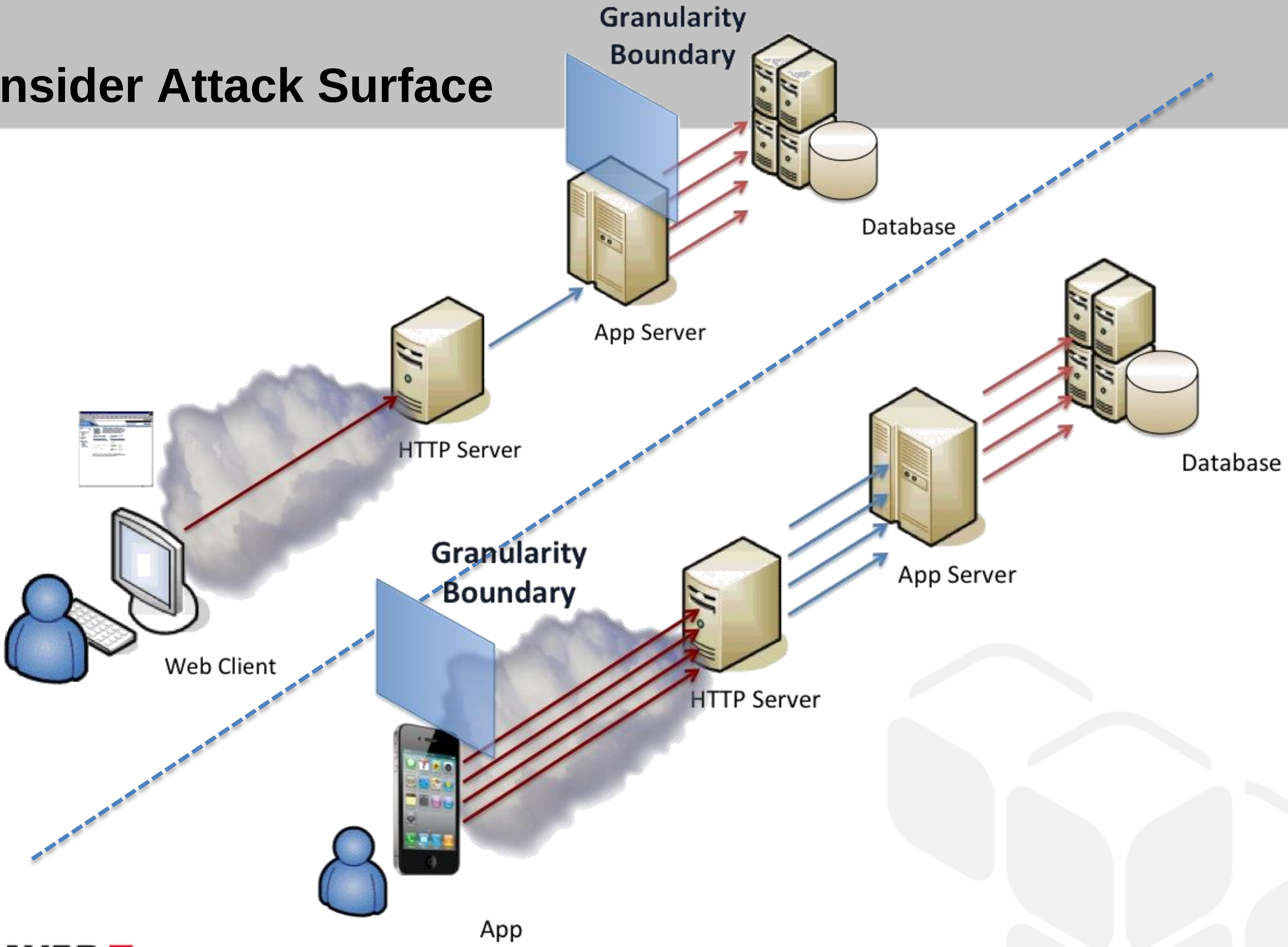
APIs Are A More Direct Conduit

Often:

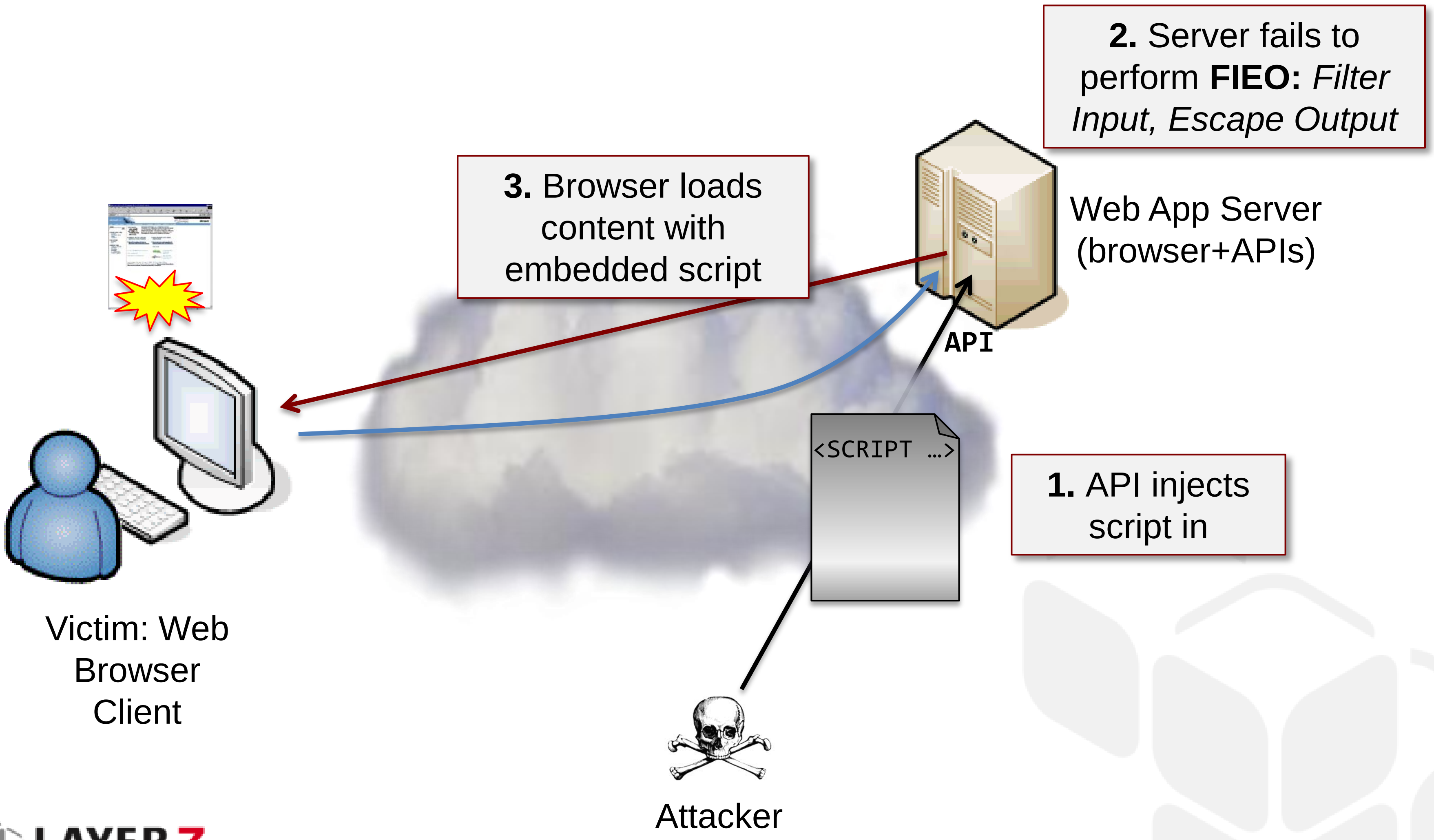
- Self-documenting
- Closely mapped to object space



Consider Attack Surface

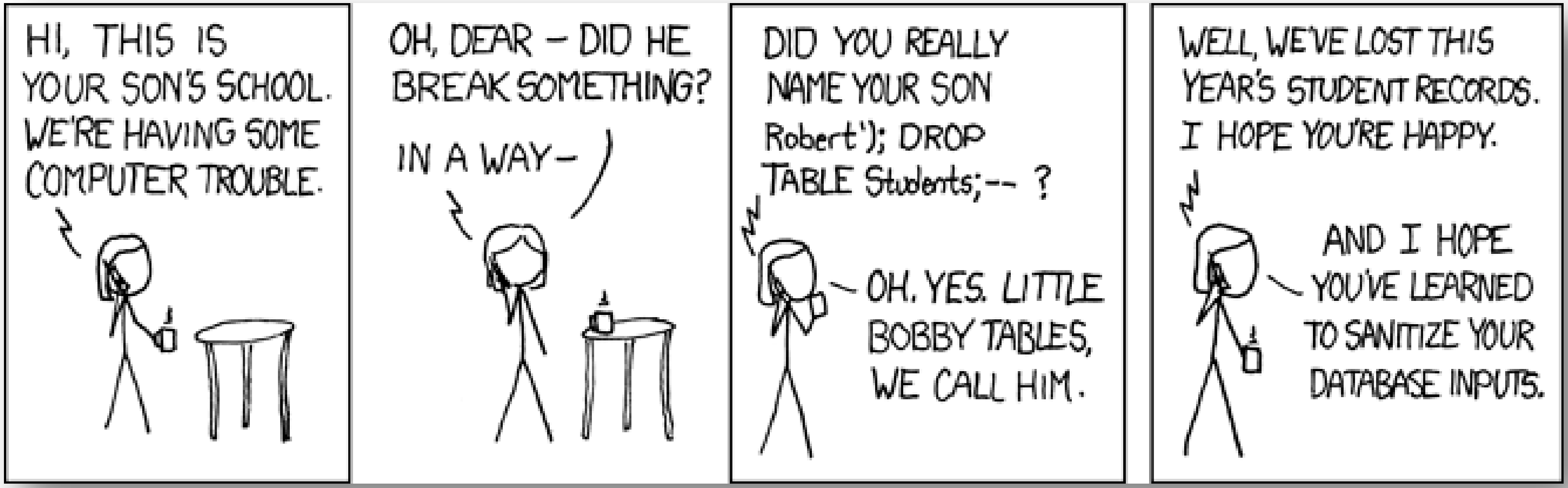


Script Insertion is Just One Potential Exploit


2. Server fails to perform **FIEO:** *Filter Input, Escape Output*
3. Browser loads content with embedded script
Web App Server (browser+APIs)
API
<SCRIPT …>
1. API injects script in
Victim: Web Browser Client
Attacker

SQL Injection is Another

Exploits of a Mom


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Source: https://xkcd.com/327/

LAYER 7 TECHNOLOGIES

Mitigation Strategy

- Rigorous validation of user supplied input
  - Stronger typing
  - Sets and ranges
  - Avoid auto-generated schemas that make everything a string
- Use schema validation
  - XML Schema, RELAX-NG, Schematron
    - Please no DTDs!
  - JSON schema validation
  - WADL’s second coming

Mitigation Strategy *(cont.)*

- Regex scanning for signatures
- Tune scanning for the API
  - Sometimes `SELECT` is OK
- Virus scanning of attachments
  - Don’t forget B64’d message content
- Library, service, or gateway solutions
  - Decoupled security
  - What are the tradeoffs?

Mitigation Strategy

- Whitelist tags if you can (i.e. where the validation space is small and concise)
  - Not always practical
  - (Note that I’m referring to whitelisting tags not IPs.)
- Blacklist dangerous tags like `<SCRIPT>`
- Always perform FIEO (Filter Input, Escape Output)
- Learn more: http://xssed.com

Problem Area #2: *Identity*

- We had it surprisingly good in the Web world
  - Browser session usually tied to human
  - Dealing with one identity is not so tough
    - Security tokens abound, but solutions are mature
      - Username/pass, x.509 certs, multi-factor, Kerberos, SAML, etc
  - APIs rapidly becoming more difficult
    - Non-human entities
    - Multiple layers of relevant identities
      - Me, my attributes, my phone, my developer, my provider…

API Keys

"An **application programing interface key** (API key) is a code generated by websites that allow users to access their application programming interface. API keys are used to track how the API is being used in order to prevent malicious use or abuse of the terms of service.

API keys are based on the UUID system to ensure they will be unique to each user."

Source: wikipedia http://en.wikipedia.org/wiki/Application_programming_interface_key

For Example:

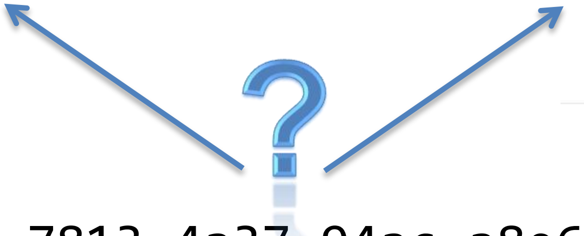
```
GET http://example.layer7.com/services/staff
    ?APIKey=15458617-7813-4a37-94ac-a8e6da6f6405
```

Seriously? WTF.

How Does An API Key Map To Identity?

A person?

Or an app?



15458617-7813-4a37-94ac-a8e6da6f6405

It is entirely inconsistent.

The Identity Profile

- As applications become more complex, increasingly, there is need to support a greater number of claims (multiple identity profiles)
  - Application ID
  - User ID
  - Device ID
- User attributes may include to …
  - Roles
  - Geo location
  - IP
  - User agent
  - Time of day
  - etc.

Where Did API Keys Come From?

- API keys came from Google APIs like maps, early Yahoo APIs, early Twitter APIs etc.
  - Originally meant for loose, non-authoritative tracking
  - Rate limits, approximate usage profiling
- Google geocoding v3.0 API deprecates API keys
  - Uses IP instead to track and throttle
  - This has its own set of problems
    - IP address spoofing
    - Problems with legitimate clients like cloud servers
- Google Premier uses a public client_id and requires signed URLs
  - (Strips domain leaving only path and query parameters)

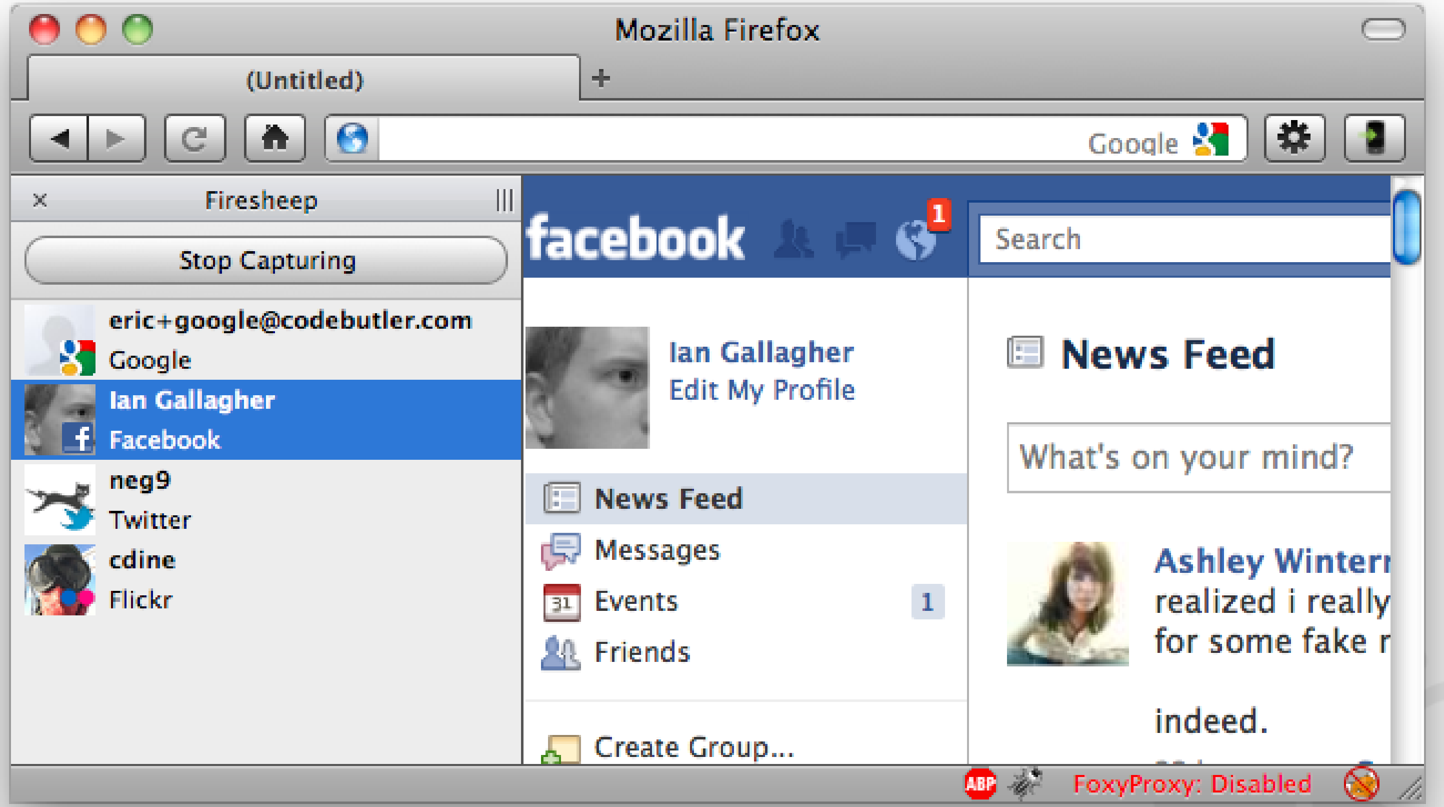
The Real Issue Is That This Is Ad-hoc Sessioning

- This is a web developer cultural issue
  - On the web, session tokens are rarely secured
    - **Step 1:** Sign in with SSL
    - **Steps 2 to n:** Track session using cookie without SSL
- There are two bad things going on here:
  1. No protection of security token
  2. No binding of token to message content

→ These two things are closely related

- In 2012, this is a huge problem…

One Word: *Firesheep*



Source: http://codebutler.com/firesheep

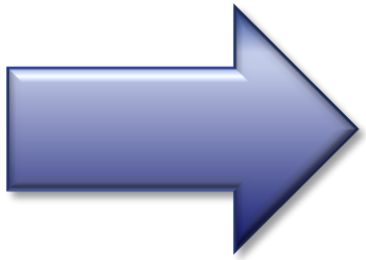
In the post-Firesheep world, unprotected session tracking should never be tolerated …

Bottom Line: The API Key Was Never Meant To Be Authoritative

- Strange hybrid of HTTP’s USER-AGENT and sessioning
- OK only for general tracking
- Anything that matters should use real security tokens
- Anything that matters == anywhere where identity is important:
  - APIs that provide access to sensitive data
  - APIs that change things that matter
  - APIs that charge for use
  - etc.

Twitter Gets It

API Keys,
Basic Authentication



Will OAuth survive adolescence?

- Complexity and confusion
- Interop is a nightmare right now
- Enterprise token management
  - Beyond the timeout
  - Lifecycle, revocation, distribution, etc.
  - Talk, but not a lot of action
- SSL everywhere
  - Protect bearer tokens
- Phishing vulnerabilities
  - Server authentication and loss of trust in the CA infrastructure

Mitigation

- **Protect the tokens!**
- HTTPS everywhere
  - This is another web design cultural issue
  - It’s just not that expensive any more
- Need HTTP Strict Transport Security (HSTS) for APIs
  - http://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security

Problem Area #3: *Cryptography and PKI*

- Cryptography is reasonably mature on the web
  - Surprisingly limited use patterns
  - SSL/TLS
  - Very little tough PKI (like client-side)
- So what’s wrong with APIs?

It’s Like We Forgot Everything We Knew

- Emailing keys
  - API keys, shared secrets, etc.
- Bad storage schemes
  - Security through obscurity
  - Toy ciphers
- No life cycle management
  - Limits on use
  - Time limits
  - Revocation
  - Audit

The Issues

- Key management
  - Especially across farms
- Nobody takes web PKI seriously
  - CRLs and OCSP aren't much good in the browser world
    - Fail open – seriously
- CA trust breakdown
  - Comodo fraud in March 2011

The Issues *(cont.)*

- Cipher suite restrictions
  - Avoiding downgrades
- Client-side certificate authentication is hard
- The alternatives for parameter confidentiality and/or integrity are hard:
  - XML encryption is still there
    - Not for the faint of heart
  - OAuth 1.0 gave you parameter signing
    - Only optional in 2.0

Mitigations

- Use real PKI
  - I know it's painful
- Use HSMs to protect keys that really matter
- Otherwise use real key material protection schemes
  - PKCS #12, etc.
  - Libraries abound

Mitigations

- You must do CRLs and OCSP for APIs
- Google maintains a certificate registry: http://googleonlinesecurity.blogspot.com/2011/04/improving-ssl-certificate-security.html
- Google safe browsing API: http://code.google.com/apis/safebrowsing/developers_guide_v2.html
  - Blacklist of phishing and malware sites
- DANE and DNSSEC

Where Does This All Leave Us?

- SOAP, the WS-* stack dealt with much of this very rigorously
  - But it was just too hard.
- We need to learn from this, but make it easier to implement
- Here’s how…

How Do I Apply This Today?

- Use SSL for all API transactions
  - Hides many sins
    - Confidentiality, integrity, replay, binding token+message, server authentication, etc.
- Use real PKI
  - Yes, it’s hard
  - But you can’t skimp here
- Use OAuth for distributed authentication
- Use real frameworks, don’t reinvent
  - They exist for most languages
- Consider gateways to encapsulate security

Thank You!

Layer 7 Technologies
Stephen Mak
Email: smak@layer7tech.com
Website: www.layer7tech.com