

# Defending Computer Networks

## *Lecture 19: Proxies and XSS*

Stuart Staniford

Adjunct Professor of Computer Science

# Logistics

- HW4 written, on website today
- Piazza still pending
  - Note email to my cornell address unreliable
  - Use backup cc: stuart – at – earlywarn – dot – org

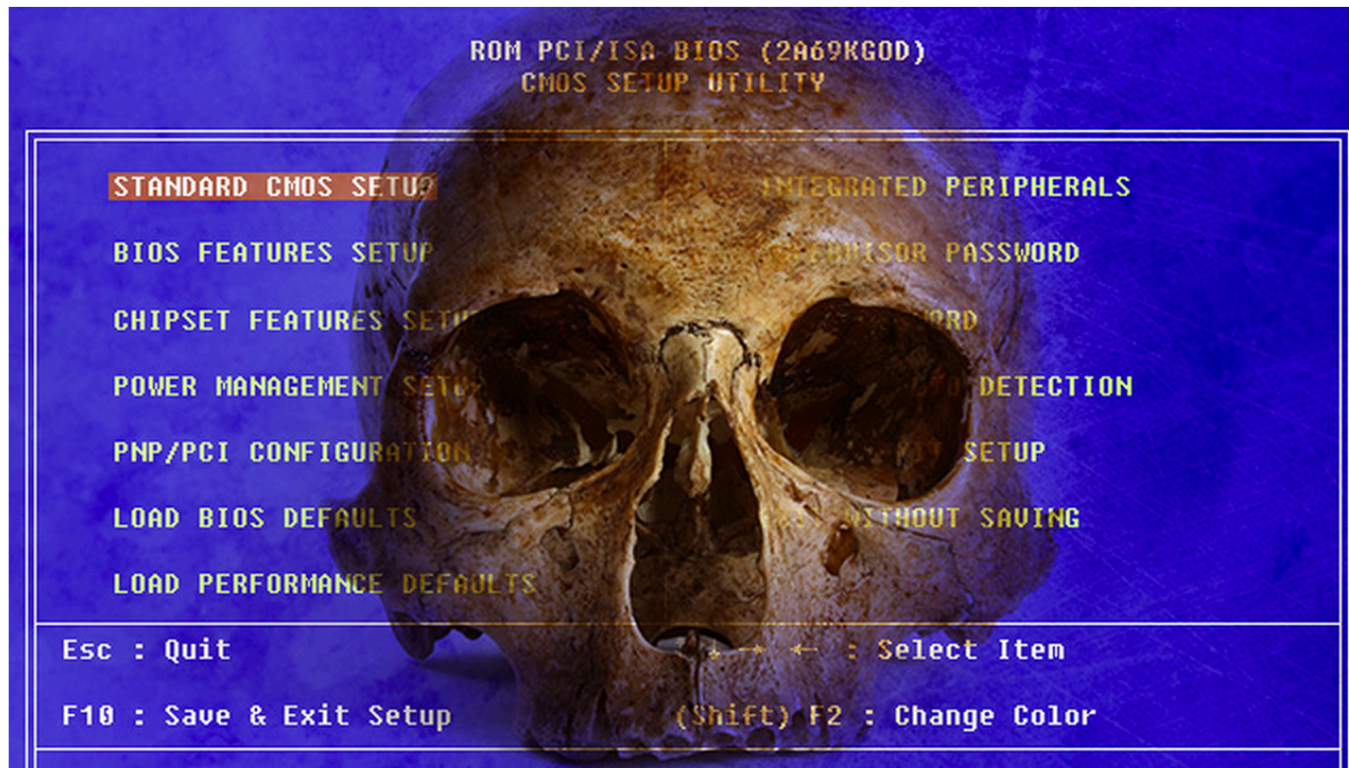
# Latest News

## Meet “badBIOS,” the mysterious Mac and PC malware that jumps airgaps


Like a super strain of bacteria, the rootkit plaguing Dragos Ruiu is omnipotent.

by Dan Goodin - Oct 31 2013, 10:07am EDT

BLACK HAT HACKING 583



# Dragos Ruiu

[Create account](#)  [Log in](#)



**WIKIPEDIA**  
The Free Encyclopedia

- [Main page](#)
- [Contents](#)
- [Featured content](#)
- [Current events](#)
- [Random article](#)
- [Donate to Wikipedia](#)
- [Wikimedia Shop](#)

- ▼ [Interaction](#)
  - [Help](#)
  - [About Wikipedia](#)
  - [Community portal](#)
  - [Recent changes](#)
  - [Contact page](#)

► [Toolbox](#)

► [Print/export](#)

Article

[Talk](#)

Read

[Edit](#)

[View history](#)

Search



## Pwn2Own

From Wikipedia, the free encyclopedia

**Pwn2Own** is a [computer hacking](#) contest held annually at the [CanSecWest security conference](#), beginning in 2007.<sup>[1]</sup> Contestants are challenged to [exploit](#) widely used [software](#) and mobile devices with [previously unknown vulnerabilities](#). Winners of the contest receive the device that they exploited, a cash prize, and a "Masters" jacket celebrating the year of their win. The name "Pwn2Own" is derived from the fact that contestants must "[pwn](#)" or hack the device in order to "own" or win it. The Pwn2Own contest serves to demonstrate the vulnerability of devices and software in widespread use while also providing a checkpoint on the progress made in security since the previous year.

### Contents [\[hide\]](#)

- [Origins](#)
- [Contest 2007](#)
  - [2.1 Rules](#)
  - [2.2 Outcome](#)
- [Contest 2008](#)
  - [3.1 Rules](#)
  - [3.2 Outcome](#)
- [Contest 2009](#)
  - [4.1 Web Browser Rules](#)

# Assigned Reading

- [http://www.cert.org/tech\\_tips/malicious\\_code\\_mitigation.html](http://www.cert.org/tech_tips/malicious_code_mitigation.html)

# Where We Are in Syllabus

## Rough Lecture Syllabus:

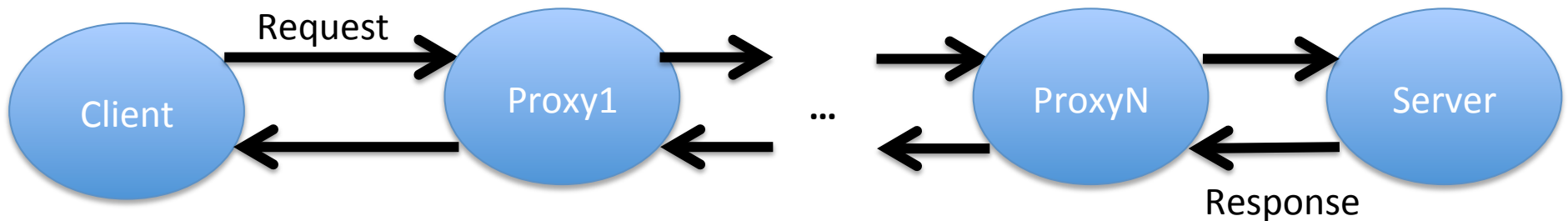
- ✓ 1. The technical nature of software vulnerabilities and techniques used for exploiting them.
- ✓ 2. The pressures of commercial software development, and why firms very rarely produce secure software, even though they should.
- ✓ 3. Basics of monitoring a network, intro/refresher on TCP/IP. Switches, wireless access devices, routers.
- ✓ 4. Network reconnaissance techniques – ping sweeps, port scans, etc.
- ✓ 5. Algorithms for detecting port scans on the network.
- ✓ 6. Firewalls and network segmentation as a defense against inbound attacks.
- ✓ 7. Detecting exploits with string matching approaches (Snort and similar).
- ✓ 8. Network layer approaches to evading detection.
- ✓ 9. Large scale attacks – worms and distributed denial of service.
- ☞ 10. HTTP attacks as a way around the firewall. Drive-by downloads and social engineering.
- ☞ 11. Defending against HTTP attacks. Web-proxies, in-browser defenses, anti-virus systems.
  12. SMTP attacks – spear-phishing, and defenses against it.
  13. HTTPS: Encryption and virtual private networks as a means to maintain confidentiality.
- ☞ 14. The modern enterprise network: what a large-scale network looks like, and emerging trends affecting it (BYOD, cloud).
  15. Legal and ethical issues in defending networks.

# Main Goals for Today

- Web proxies
- Cross-site Scripting

# Web Proxies

- HTTP designed to support chains of proxies:



- Browser/OS has support to designate a proxy



# Some HTTP Features for Proxies

- If-Modified-Since: <date>
  - Request side header
  - Allows a 304 Not Modified response
- If-Match: <entity-tag>
- Cache-Control: no-cache (etc)
- Via: <proxy>

# URL Blacklists

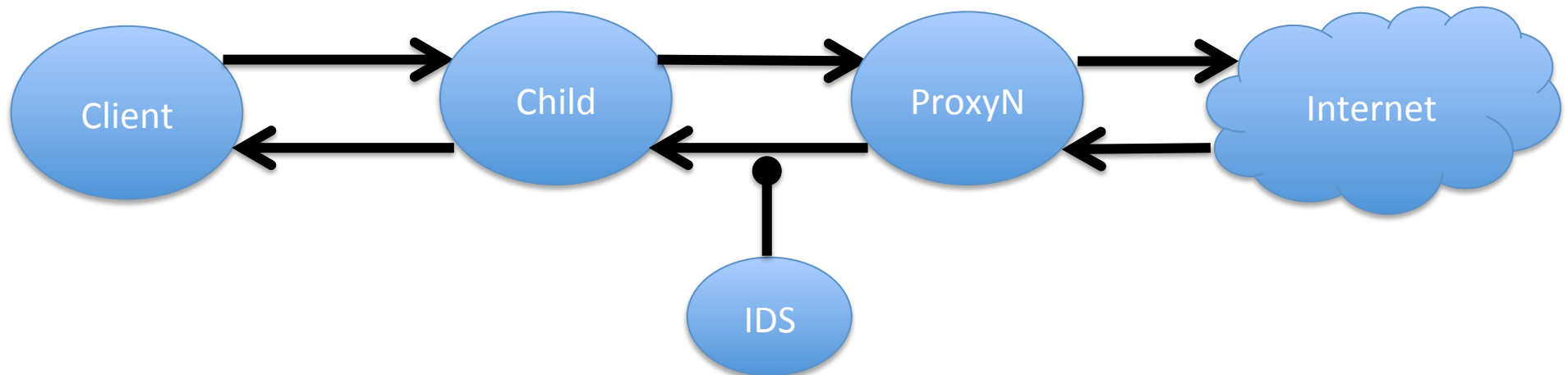
- List of “bad” urls
  - Known malicious
    - Malware, etc
    - Google safe browsing is most famous
  - Productivity problem categories
    - Adult
    - Gambling
    - Social Media
    - Hobby
    - Sports
    - News
  - Uncategorized
    - Blocking this avoids many problems, but also FPs

# Building a URL Blacklist

- Build a big farm of clients (eg in VMs)
- Crawl the web
- Try to get infected
- Note the bad URLs
- If you were the bad guys, what would you do?

# Reasons for Client-side proxy chains

- Acquisitions
  - When BigCo acquires SmallCo
  - Easiest thing is make SmallCo proxy point to BigCo proxy
  - Don't have to change settings on all SmallCo computers
- Proxy Sandwich
  - Allow for monitoring between child and parent



# X-Forwarded-For

- When there is a client-side proxy
  - Anything on Internet side will not see original IP address of client
  - If this is desirable,
    - X-forwarded-for: <ip1>, <ip2>, ...
    - Records the chain of IP addresses (original client and proxies along the way).
- In proxy sandwich architecture, often see
  - Child proxy adds X-forwarded-for
  - Parent proxy removes it again

# Cross-Site Scripting




Rank	Score	ID	Name
[1]	93.8	<a href="#">CWE-89</a>	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	83.3	<a href="#">CWE-78</a>	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[3]	79.0	<a href="#">CWE-120</a>	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	77.7	<a href="#">CWE-79</a>	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[5]	76.9	<a href="#">CWE-306</a>	Missing Authentication for Critical Function
[6]	76.8	<a href="#">CWE-862</a>	Missing Authorization
[7]	75.0	<a href="#">CWE-798</a>	Use of Hard-coded Credentials
[8]	75.0	<a href="#">CWE-311</a>	Missing Encryption of Sensitive Data
[9]	74.0	<a href="#">CWE-434</a>	Unrestricted Upload of File with Dangerous Type
[10]	73.8	<a href="#">CWE-807</a>	Reliance on Untrusted Inputs in a Security Decision
[11]	73.1	<a href="#">CWE-250</a>	Execution with Unnecessary Privileges
[12]	70.1	<a href="#">CWE-352</a>	Cross-Site Request Forgery (CSRF)
[13]	69.3	<a href="#">CWE-22</a>	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[14]	68.5	<a href="#">CWE-494</a>	Download of Code Without Integrity Check
[15]	67.8	<a href="#">CWE-863</a>	Incorrect Authorization
[16]	66.0	<a href="#">CWE-829</a>	Inclusion of Functionality from Untrusted Control Sphere
[17]	65.5	<a href="#">CWE-732</a>	Incorrect Permission Assignment for Critical Resource
[18]	64.6	<a href="#">CWE-676</a>	Use of Potentially Dangerous Function
[19]	64.1	<a href="#">CWE-327</a>	Use of a Broken or Risky Cryptographic Algorithm
[20]	62.4	<a href="#">CWE-131</a>	Incorrect Calculation of Buffer Size
[21]	61.5	<a href="#">CWE-307</a>	Improper Restriction of Excessive Authentication Attempts
[22]	61.1	<a href="#">CWE-601</a>	URL Redirection to Untrusted Site ('Open Redirect')
[23]	61.0	<a href="#">CWE-134</a>	Uncontrolled Format String
[24]	60.3	<a href="#">CWE-190</a>	Integer Overflow or Wraparound
[25]	59.9	<a href="#">CWE-759</a>	Use of a One-Way Hash without a Salt







# Still a Live Issue

## Facebook Login Page hacked through XSS by Mauritania Attacker

Posted by: HNBulletin in Facebook, Mauritania Attacker, News, XSS ⌚ June 2, 2013 💬 2 Comments

2

 Share

 Like { 2 }  Tweet { 0 }  Share  Submit  submit  +1 { 9 }



**Sign Up**  
It's free and always will be.

HACKED BY MAURITANIA ATTACKER [Change](#)

Birthday:

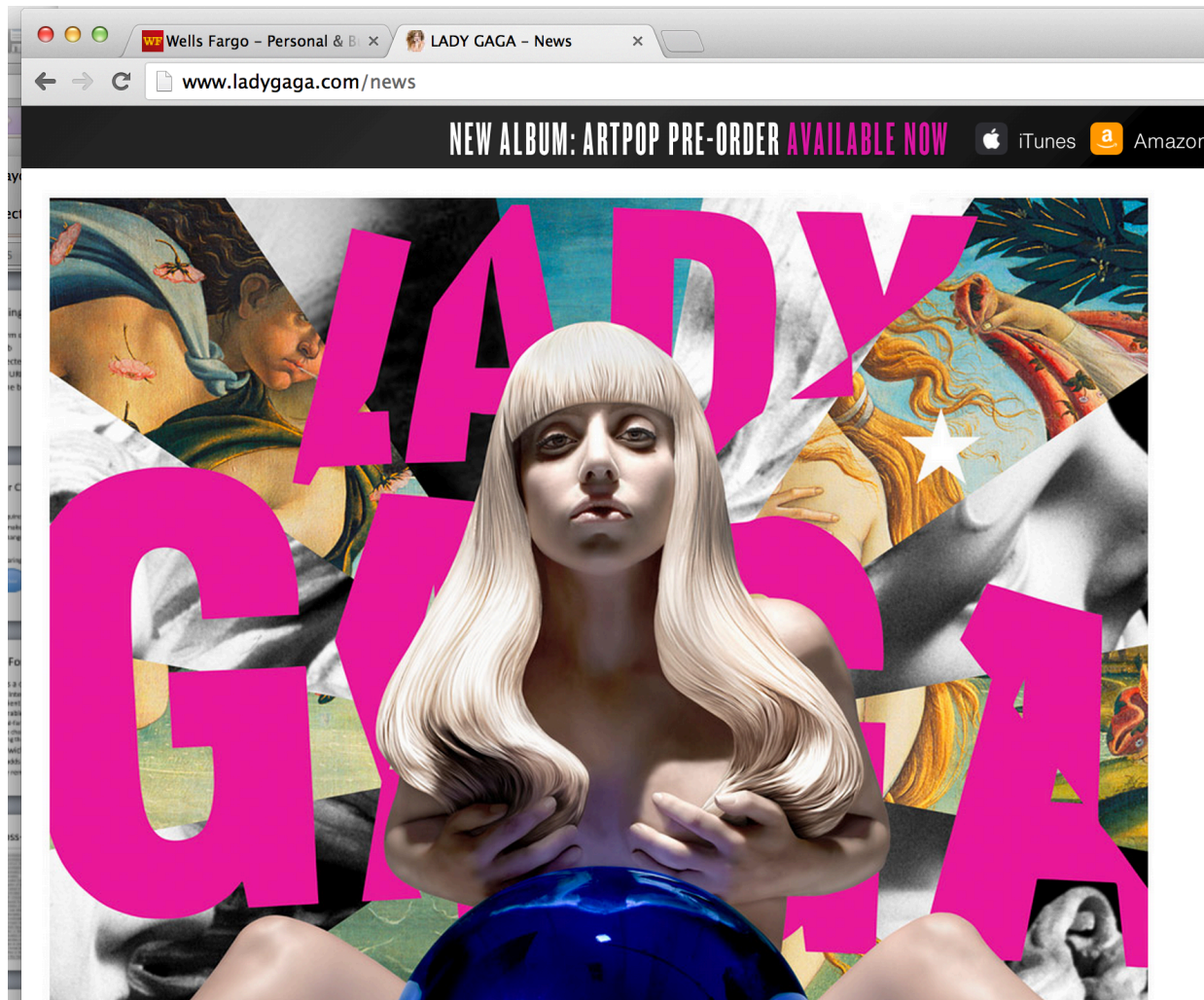
Month:  Day:  Year:  [Why do I need to provide my birthday?](#)

Female  Male

Founder of *Anonghost* team "Mauritania Attacker" found XSS Vulnerability in *Facebook.com* which adds their own message (**HACKED BY MAURITANIA ATTACKER**) in the Facebook Login Page and we also checked that it is still working.

# Same Origin Policy

- When can a piece of js access a DOM?





# Same Origin Policy

- Principle enforced by browser is:
  - Protocol, host, and port must all match

Compared URL	Outcome	Reason
<b>http://www.example.com/dir/page2.html</b>	Success	Same protocol and host
<b>http://www.example.com/dir2/other.html</b>	Success	Same protocol and host
<b>http://username:password@www.example.com/dir2/other.html</b>	Success	Same protocol and host
http://www.example.com: <b>81</b> /dir/other.html	Failure	Same protocol and host but different port
<b>https://</b> www.example.com/dir/other.html	Failure	Different protocol
http:// <b>en</b> .example.com/dir/other.html	Failure	Different host
http:// <b>example.com</b> /dir/other.html	Failure	Different host (exact match required)
http:// <b>v2</b> .www.example.com/dir/other.html	Failure	Different host (exact match required)
http://www.example.com: <b>80</b> /dir/other.html	Don't use	Port explicit. Depends on implementation in browser.

So ladygaga.com <script>s shouldn't be able to talk to wells Fargo.com

# Form Generation

- [http://www.w3schools.com/html/html\\_forms.asp](http://www.w3schools.com/html/html_forms.asp)
  - Especially examine the submit button form
  - Use the submit button
  - Examine the url with parameters
  - Examine the generated output html source
  - What is the server code doing here?
  - Try inputting `<i>blah</i>`

# Website Login

The screenshot shows a web browser window with the address bar displaying <https://www.wellsfargo.com>. The page features the Wells Fargo logo and a navigation menu with categories: Personal, Small Business, Commercial, Financial Education, and A. Below the navigation menu, there are links for Banking, Loans and Credit, Insurance, Investing and Retirement, and Wealth Management. The main content area is divided into three sections: a login form on the left, a central image of three smiling young adults, and a purple promotional banner on the right. The login form includes a 'View Your Accounts' header, a dropdown menu for 'Account Summary', input fields for 'Username' and 'Password', a 'Go' button, and links for 'Username / Password Help', 'Try Bill Pay on the go. Learn more.', and 'Your Privacy and Security'. The promotional banner on the right has the text 'No payments while in school' and 'Get the funds you need to pay for', with a 'Start Now' button.

Wells Fargo – Personal & Business

[Sign Up](#) [Customer Service](#) [ATMs/Locations](#) [Español](#)

**WELLS FARGO**

**Personal** Small Business Commercial Financial Education A

Banking Loans and Credit Insurance Investing and Retirement Wealth Management

**View Your Accounts**

Account Summary

Username

Password

[Username / Password Help](#)

Try Bill Pay on the go. [Learn more.](#)  
[Your Privacy and Security](#)

No payments while in school

Get the funds you need to pay for

Fraud Information Home Lending Retirement Borrowing and Banking M

# How Does Bank Maintain State?

The screenshot shows a web browser window with the address bar containing the URL `https://online.wellsfargo.com/das/cgi-bin/session.cgi?screenid=SIGNON_PORTAL_PAUSE`. The page header includes the Wells Fargo logo and navigation links: [Sign Off](#), [Home](#), [Locations](#), [Contact Us](#), and [Online Security Guarantee](#). The main navigation menu contains: [Accounts](#), [Transfers & Payments](#), [Brokerage](#), [Account Services](#), [Messages & Alerts](#), [Online Solutions](#), and [Open an Account](#). A sub-menu under [Accounts](#) includes [Account Summary](#), [Account Activity](#), [Money Map](#), and [Statements & Documents](#). The page content shows the user's last sign-on date as November 01, 2013, and a link to [Buy Wells Fargo Visa® Gift Cards in](#). Below the navigation, there are two sections: **Communications Summary** and **Cash Accounts**.

**Communications Summary**

Messages & Alerts: 0 new messages since you last visited your [Inbox](#)

**Cash Accounts**

Account	Available Balance	Relate
...	...	...

# Cookies

- RFC 6265
  - obsoletes RFCs 2965 and 2109
- Mainly defines two HTTP headers
  - Set-Cookie:
    - Server to client (browser)
    - Defines name/value/attribute of cookie
  - Cookie:
    - Client to server
    - Reports on cookies stored for that server

# Set-Cookie: Syntax

```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: name=value
Set-Cookie: name2=value2; Expires=Wed, 09 Jun 2021 10:18:14 GMT

(content of page)
```

# Cookie: Syntax

```
GET /spec.html HTTP/1.1  
Host: www.example.org  
Cookie: name=value; name2=value2  
Accept: */*
```

# Looking at HTTP Headers

- `curl -D cnn-header.txt http://www.cnn.com`
- More `cnn-header.txt`

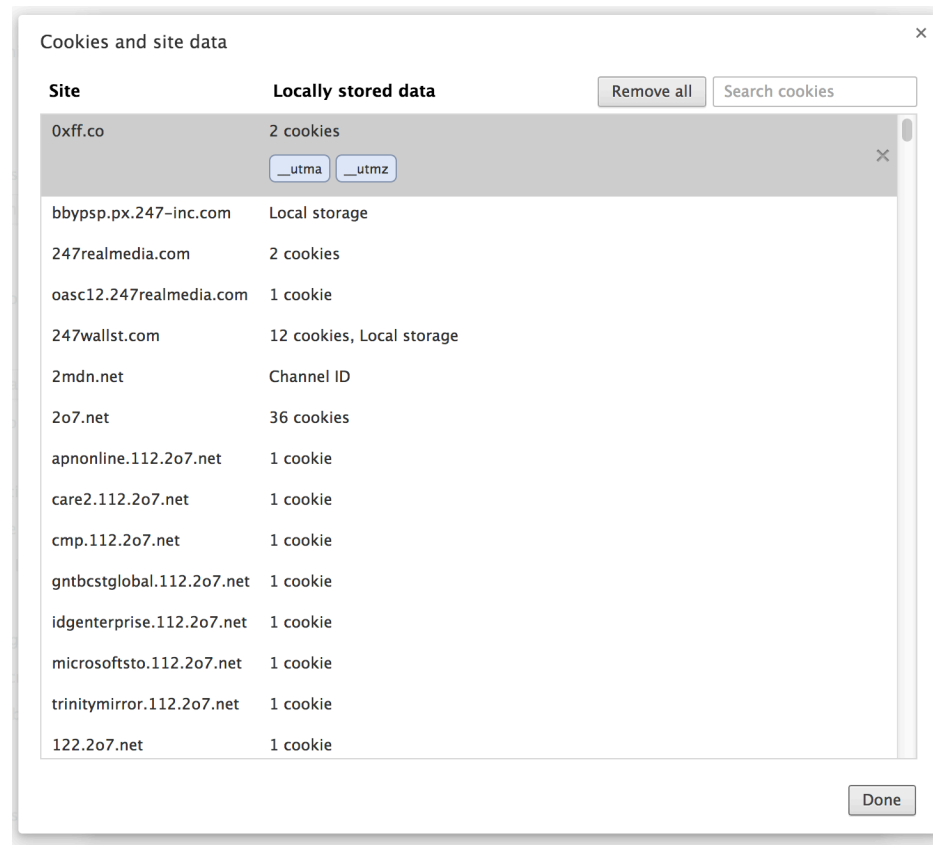


# Types of Cookie

- Session Cookie
  - No expiration set
  - Gone on browser close
- Persistent Cookie
  - Stored on disk, long-lived in browser

# Cookies In The Browser

- <https://support.google.com/chrome/answer/95647?hl=en>



# Accessing Cookies from JS

## Summary

---

Get and set the cookies associated with the current document.

## Syntax

---

```
allCookies = document.cookie;
```

*allCookies* is a string containing a semicolon-separated list of cookies (i.e. *key=value* pairs)

```
document.cookie = updatedCookie;
```

*updatedCookie* is a string of form *key=value*. Note that you can only set/update a single cookie at a time using this method.

<https://developer.mozilla.org/en-US/docs/Web/API/document.cookie>

# Putting It Together

- Elements of an XSS attack scenario
  - I use server with sensitive content (bank)
  - Bank server code that doesn't eliminate markup
  - Attacker (Lady Gaga) tricks me into visiting a link to bank,
    - but of her construction
    - while I'm logged into bank
  - Bank incorporates Lady Gaga's code into webpage
  - Now her javascript can access bank
    - with my login privileges (has my cookie)
    - Now she can steal my \$609.31!

# Ways to Deliver the URL

- Email
- In another web page
  - Link to click
  - Iframe with bank url