Announcements

• Next Wednesday’s lecture will be given by Lisa

• I won’t have my usual 3-4PM office hours next Wednesday, but will be available at the usual 3-4PM slot on Friday …
  – … as well as by appointment via email (as always)

• Reminder: first phase of Project #1 due next Wednesday by 11PM
  – The writeup has been updated for clarity, see mailing list archives for “diffs”

• Thanksgiving week: I’ll give the same lecture twice, Mon 4-5:30PM (room TBD) and Weds (usual)
Goals of Today’s Lecture

- Finish discussion of the workings of DNS
- DNS security analysis

- Applications in general …
- … and Email in particular

```
unix> dig +norecurse @a.root-servers.net in-addr.arpa ns

; <<>> DiG 9.3.4 <<>> +norecurse @a.root-servers.net in-addr.arpa ns
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62001
;; flags: qr aa; QUERY: 1, ANSWER: 12, AUTHORITY: 0, ADDITIONAL: 12

;; QUESTION SECTION:
in-addr.arpa.               IN   NS

;; ANSWER SECTION:
in-addr.arpa.       86400 IN   NS   G.ROOT-SERVERS.NET.
in-addr.arpa.       86400 IN   NS   H.ROOT-SERVERS.NET.
in-addr.arpa.       86400 IN   NS   I.ROOT-SERVERS.NET.
in-addr.arpa.       86400 IN   NS   K.ROOT-SERVERS.NET.
in-addr.arpa.       86400 IN   NS   L.ROOT-SERVERS.NET.
in-addr.arpa.       86400 IN   NS   M.ROOT-SERVERS.NET.
in-addr.arpa.       86400 IN   NS   A.ROOT-SERVERS.NET.
in-addr.arpa.       86400 IN   NS   B.ROOT-SERVERS.NET.
in-addr.arpa.       86400 IN   NS   C.ROOT-SERVERS.NET.
in-addr.arpa.       86400 IN   NS   D.ROOT-SERVERS.NET.
in-addr.arpa.       86400 IN   NS   E.ROOT-SERVERS.NET.
in-addr.arpa.       86400 IN   NS   F.ROOT-SERVERS.NET.
```
unix> dig +nosecure @a.root-servers.net -x 64.236.24.12

;; QUESTION SECTION:
;12.24.236.64.in-addr.arpa. IN PTR

;; AUTHORITY SECTION:
64.in-addr.arpa. 86400 IN NS dill.ARIN.NET.
64.in-addr.arpa. 86400 IN NS BASIL.ARIN.NET.
64.in-addr.arpa. 86400 IN NS henna.ARIN.NET.
64.in-addr.arpa. 86400 IN NS indigo.ARIN.NET.
64.in-addr.arpa. 86400 IN NS epazote.ARIN.NET.
64.in-addr.arpa. 86400 IN NS figwort.ARIN.NET.
64.in-addr.arpa. 86400 IN NS chia.ARIN.NET.

(no ADDITIONAL section)

;; Query time: 93 msec
;; SERVER: 198.41.0.4#53(198.41.0.4)
;; WHEN: Thu Sep 20 23:50:49 2007
;; MSG SIZE  rcvd: 194

unix> dig +nosecure @dill.arin.net -x 64.236.24.12

;; QUESTION SECTION:
;12.24.236.64.in-addr.arpa. IN PTR

;; AUTHORITY SECTION:
236.64.in-addr.arpa. 86400 IN NS dns-02.atdn.net.
236.64.in-addr.arpa. 86400 IN NS dns-01.atdn.net.

unix> dig +nosecure @dns-02.atdn.net -x 64.236.24.12

;; QUESTION SECTION:
;12.24.236.64.in-addr.arpa. IN PTR

;; ANSWER SECTION:

;; AUTHORITY SECTION:
24.236.64.in-addr.arpa. 3600 IN NS dns-02.atdn.net.
24.236.64.in-addr.arpa. 3600 IN NS dns-01.atdn.net.

;; ADDITIONAL SECTION:
dns-01.atdn.net. 3600 IN A 64.12.51.136
dns-02.atdn.net. 3600 IN A 205.188.157.236
Inserting Resource Records into DNS

• Example: just created startup “FooBar”
• Get a block of address space from ISP
  – Say 212.44.9.128/25
• Register foobar.com at Network Solutions (say)
  – Provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
  – Registrar inserts RR pairs into the com TLD server:
    o (foobar.com, dns1.foobar.com, NS)
    o (dns1.foobar.com, 212.44.9.129, A)
• Put in your (authoritative) server dns1.foobar.com:
  – Type A record for www.foobar.com
  – Type MX record for foobar.com

Setting up foobar.com, con’t

• In addition, need to provide reverse PTR bindings
  – E.g., 212.44.9.129 → dns1.foobar.com
• Normally, these would go in 9.44.212.in-addr.arpa
• Problem: you can’t run the name server for that domain. Why not?
  – Because your block is 212.44.9.128/25, not 212.44.9.0/24
  – And whoever has 212.44.9.0/25 won’t be happy with you owning their PTR records
• Solution: ISP runs it for you
  – Now it’s more of a headache to keep it up-to-date :-(

Security Analysis of DNS

- What security issues does the design & operation of the Domain Name System raise?
- Degrees of freedom:

<table>
<thead>
<tr>
<th>16 bits</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</td>
<td>Flags</td>
</tr>
<tr>
<td># Questions</td>
<td># Answer RRs</td>
</tr>
<tr>
<td># Authority RRs</td>
<td># Additional RRs</td>
</tr>
</tbody>
</table>

Questions (type, class, domain name)
Answers (variable # of resource records)
Authority (variable # of resource records)
Additional information (variable # of resource records)

Security Problem #1: Starbucks

- As you sip your latte and surf the Web, how does your laptop find google.com?
- Answer: it asks the local name server per Dynamic Host Configuration Protocol (DHCP) …
  - … which is run by Starbucks or their contractor
  - … and can return to you any answer they please
  - … including a “man in the middle” site that forwards your query to Google, gets the reply to forward back to you, yet can change anything they wish in either direction
- How can you know you’re getting correct data?
  - Today, you can’t. (Though if site is HTTPS, that helps)
  - One day, hopefully: DNSSEC extensions to DNS
Security Problem #2: Cache Poisoning

- Suppose you are a Bad Guy and you control the name server for foobar.com. You receive a request to resolve www.foobar.com and reply:

```
;; QUESTION SECTION:
;www.foobar.com. IN A

;; ANSWER SECTION:
www.foobar.com. 300 IN A 212.44.9.144

;; AUTHORITY SECTION:
foobar.com. 600 IN NS google.com.

;; ADDITIONAL SECTION:
google.com. 5 IN A 212.44.9.155
```

Evidence of the attack disappears 5 seconds later!

A foobar.com machine, not google.com

Cache Poisoning, con’t

- Okay, but how do you get the victim to look up www.foobar.com in the first place?
- Perhaps you connect to their mail server and send
  - HELO www.foobar.com
  - Which their mail server then looks up to see if it corresponds to your source address (anti-spam measure)

- Note, with compromised name server we can also lie about PTR records (address → name mapping)
  - E.g., for 212.44.9.155 = 155.44.9.212.in-addr.arpa return google.com (or whitehouse.gov, or whatever)
    - If our ISP lets us manage those records as we see fit, or we happen to directly manage them
Cache Poisoning, con’t

- Suppose Bad Guy is at Starbuck’s and they can sniff (or even guess) the identification field the local server will use in its next request:

- They:
  - Ask local server for a (recursive) lookup of google.com
  - Locally spoof subsequent reply from correct name server using the identification field
  - Bogus reply arrives sooner than legit one

- Local server duly caches the bogus reply!
  - Now: every future Starbuck customer is served the bogus answer out of the local server’s cache
    - In this case, the reply uses a large TTL

Security Summary

- DNS lacks authentication
  - Can’t tell if reply comes from the correct source
  - Can’t tell if correct source tells the truth
  - Malicious source can insert extra (mis)information
  - Malicious bystander can spoof (mis)information

- Playing with caching lifetimes adds extra power to attacks
5 Minute Break

Questions Before We Proceed?

Application Protocols
Application-Layer Protocols

• The end systems involved in any network application have to decide how they understand the messages they exchange

• This is done by explicitly defining a protocol
  – Types of messages (e.g., requests and responses)
  – Message syntax (e.g., fields, and how to delineate)
  – Semantics of the fields (i.e., meaning of the information)
  – Rules for when and how a process sends messages

• This application-layer protocol is sent on top of a transport protocol (layering) …

• … which in turn is sent on top of a network-layer protocol (IPv4, for us)

Applications vs. Application-Layer Protocols

• Application-layer protocol is just one piece
  – Defining how the end hosts communicate

• Example: World Wide Web
  – HyperText Transfer Protocol is the protocol
  – But the Web includes other components, such as document formats (HTML), Web browsers, servers, …

• Example: electronic mail
  – Simple Mail Transfer Protocol (SMTP) is one of the protocols
  – But e-mail includes other components, such as mail servers, user mailboxes, mail readers
A Wide Range of Application Styles

• Underlying transport:
  – TCP (Web, Email), UDP (DNS, Network Time), both (NFS, Skype)

• Connection structure:
  – Single connection (SSH), series of connections (Email, HTTP),
    concurrent connections (HTTP, FTP)
  – Control separate from data (FTP) or intermingled (HTTP, SMTP,
    Telnet)
  – Long stateful sessions (FTP, Telnet), stateless servers (HTTP)

• Application protocol format:
  – Binary (DNS, NFS), Text (SMTP, HTTP), both (SSH)

• Data encoding:
  – None (Telnet), MIME (SMTP, HTTP), XDR (RPC), custom
    compression (DNS)

Example of An Application Protocol:

Electronic Mail
E-Mail Message Format (RFC 822)

- E-mail messages have two parts
  - A header, in 7-bit U.S. ASCII text
  - A body, also represented in 7-bit U.S. ASCII text

- Header
  - Lines with “type: value”
  - “To: vern@icir.org”
  - “Subject: Go Bears!”

- Body
  - The text message
  - No particular structure or meaning
  - Need ways to deal with 8-bit text, binaries, multiple items

Multipurpose Internet Mail Extensions

- Additional headers to describe the message body
  - MIME-Version: the version of MIME being used
  - Content-Type: the type of data contained in the message
  - Content-Transfer-Encoding: how the data are encoded

- Definitions for a set of content types and subtypes
  - E.g., image with subtypes gif and jpeg
  - E.g., text with subtypes plain, html, and richtext
  - E.g., application with subtypes postscript and msword
  - E.g., multipart for messages with multiple data types

- A way to encode the data in ASCII format
  - Base64 encoding (64-character printable alphabet)
Example: E-Mail Message Using MIME

```
From: jrex@cs.princeton.edu
To: feamster@cc.gatech.edu
Subject: picture of my cat

MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

Base64 encoded data ...
JVBERi0xLjMNJeLjz9MNMSAwI
..........................
......base64 encoded data
```

E-Mail Addresses

- Components of an e-mail address
  - Local mailbox (e.g., vern or jortiz)
  - Domain name (e.g., icir.org or cs.berkeley.edu)

- Identifying the mail server for a domain
  - DNS query asking for MX records
    - Mail eXchange
      - E.g., `dig eecs.berkeley.edu mx`
      - Yields:
        - mx1.eecs.berkeley.edu, mx2.eecs.berkeley.edu,
          bcl.eecs.berkeley.edu, bc2.eecs.berkeley.edu
  - Then, regular DNS query to learn the IP address
Mail Servers and User Agents

- Mail servers
  - Always on and always accessible
  - Transferring e-mail to and from other servers

- User agents (UA)
  - Sometimes on and sometimes accessible
  - Intuitive interface for the user

SMTP Store-and-Forward Protocol

- SMTP = Simple Mail Transfer Protocol
- Messages sent through a series of servers
  - A server stores incoming messages in a queue
  - … to await attempts to transmit them to the next hop
- If the next hop is not reachable
  - The server stores the message and tries again later
- Each hop adds a “Received” header w/ its identity
  - Helpful for diagnosing problems with e-mail
Example With Received Header

Return-Path: <casado@cs.stanford.edu>
Received: from ribavirin.CS.Princeton.EDU (ribavirin.CS.Princeton.EDU [128.112.136.44])
    by newark.CS.Princeton.EDU (8.12.11/8.12.11) with SMTP id k04M5RY7Y23164
    for <jrex@newark.CS.Princeton.EDU>; Wed, 4 Jan 2006 17:05:37 -0500 (EST)
Received: from bluebox.CS.Princeton.EDU ([128.112.136.38])
    by ribavirin.CS.Princeton.EDU (3MSSMTP 4.1.0.19) with SMTP id M2006010417053607946
    for <jrex@newark.CS.Princeton.EDU>; Wed, 04 Jan 2006 17:05:36 -0500
Received: from smtp-roam.Stanford.EDU (smtp-roam.Stanford.EDU [171.64.10.152])
    by bluebox.CS.Princeton.EDU (8.12.11/8.12.11) with ESMTP id k04M5XNQ005204
    for <jrex@cs.princeton.edu>; Wed, 4 Jan 2006 17:05:35 -0500 (EST)
Received: from {192.168.1.101} (adsl-69-107-78-147.dsl.pltn13.pacbell.net [69.107.78.147])
    (authenticated bits=0)
    (version=TLSv1/SSLv3 cipher=DHE-RSA-AES256-SHA bits=256 verify=NOT);
    Wed, 4 Jan 2006 14:05:32 -0800
Message-ID: <43BC46AF.3030306@cs.stanford.edu>
Date: Wed, 04 Jan 2006 14:05:35 -0800
From: Martin Casado <casado@cs.stanford.edu>
User-Agent: Mozilla Thunderbird 1.0 (Windows/20041206)
MIME-Version: 1.0
To: jrex@CS.Princeton.EDU
CC: Martin Casado <casado@cs.stanford.edu>
Subject: Using VNS in Class
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: 7bit

Multiple Server Hops

• Typically at least two mail servers
  – Sending and receiving sides

• There may be more:
  – Separate servers for key functions
    o Spam filtering
    o Virus scanning
  – Servers that redirect the message
    o From vern@ee.lbl.gov to vern@icir.org
    o Messages to ee.lbl.gov go through extra hops
  – Electronic mailing lists
    o Mail delivered to the mailing list’s server
    o … and then the list is expanded to each recipient
Scenario: Alice Sends Message to Bob

1) Alice uses UA to compose message “to” bob@someschool.edu
2) Alice’s UA sends message to her mail server; message placed in message queue
3) Client side of SMTP opens TCP connection with Bob’s mail server
4) SMTP client sends Alice’s message over the TCP connection
5) Bob’s mail server places the message in Bob’s mailbox
6) Bob invokes his user agent to read message

Simple Mail Transfer Protocol, Cont.

• Command/response interaction
  – Commands: ASCII text
  – Response: three-digit status code and phrase

• Synchronous
  – Sender awaits response from a command
  – … before issuing the next command

• Three phases of transfer
  – Handshaking (greeting)
  – Transfer of messages
  – Closure
Sample SMTP interaction

S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: From: alice@crepes.fr
C: To: hamburger-list@burger-king.com
C: Subject: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection

Retrieving E-Mail From the Server

• Server stores incoming e-mail by mailbox
  – Based on the “From” field in the message

• Users can retrieve e-mail in variety of ways:
  – Directly by same-machine access to the mailbox
  – Via Interactive Mail Access Protocol (IMAP)
    o Supports concurrent access by multiple clients,
      server-side searchers, partial MIME fetches, multiple
      mailboxes
  – Via HTTP (Web)
    o E.g., GMail
  – Via Post Office Protocol (POP)
POP3 Protocol

Authorization phase
• Client commands:
  – user: declare username
  – pass: password
• Server responses
  – +OK
  – -ERR

Transaction phase, client:
• list: list message numbers
• retr: retrieve message by number
• dele: delete
• quit

S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on

C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off

Summary
• Domain Name System (DNS)
  – Distributed, hierarchical database
  – Distributed collection of servers
  – Caching to improve performance
  – Examine using dig utility

• Application-layer protocols
  – Rich and constantly evolving area
  – Tailoring communication to the application

• Electronic-mail protocols
  – SMTP to transfer e-mail messages
  – MIME to represent a wide variety of data formats
  – Transfers based on store-and-forward
Next Lecture

• The Web (given by Lisa)
• Reading: K&R 2.2
• First phase of Project #1 due next Weds, 11PM
• My office hours next week are just on Friday
  – And as always, by appointment via email