IT441: Network Servers & Infrastructure

CLASS 13 : 29 Nov 2004
13:30 – 16:15

Except where otherwise noted, these slides are © copyright Charles Snow, 2004. Permission to copy these slides is granted so long as each slide contains the copyright notice “© C. M. Snow 2004” appears on each page and the copies are made freely available, i.e., not sold. Students are advised that they should not use University consumables (ink, toner, paper) to produce printed versions of these slides. Rights for all other use of this material are reserved by the author; permission for any other use must be obtained in writing from the author.

Last Time

- Internet service: WWW
- 802.11 wireless
- Bluetooth
- services discovery, agents, ontologies
- pervasive computing
This Time

- RSS
- open source
- local distribution

Getting The Word Out

- suppose you have a special interest in \( x \)
  - e.g., news, sports, technical updates, AV equipment
- how do you stay current?
  - discover/learn which web sites have content
  - visit them often
- sites might prefer to be able to push their wares?
  - member subscription; they send email
  - specialized software ‘agent’ goes mining web site
Getting The Word Out

www site A

www site B

www site C

A Newer Model

www site A

www site B

www site C

www site D

web site D 'knows' a little about content on A, B, and C. you can see summaries and entire articles from A..D just by visiting D
How Build This?

- use specialized content agents
  - like crawlers, but targeted
- content-providing sites provide API for content access
How Build This?

- use specialized content agents
  - like crawlers, but targetted
- content-providing sites provide API for content access
- content-providing sites provide data dumps from their content

A Better Way To Build This

- use standardized ‘blurb’ format embodied in a technology called RSS
  - Rich Site Summary
  - a.k.a. Really Simple Syndication
- like syndicating: publish material to some number of locations
  - e.g., like comics in newspapers, some TV shows
- RSS uses XML
An RSS Item

- RSS provides a set of **items** within a **channel** to interested readers
- an item looks like:

  ```xml
  <item>
  <title>RSS Resources</title>
  <link>http://www.webreference.com/authoring/languages/xml/rss/</link>
  <description>
  Defined in XML, the Rich Site Summary (RSS) format has quietly become a dominant format for distributing headlines on the Web. Our list of links gives you the tools, tips and tutorials you need to get started using RSS.
  0323
  </description>
  </item>
  ```

RSS Channel

- channel provides set of items in some way related
  - e.g., most recent, same topic
  - up to 15 items per channel
- RSS element may contain at most 1 channel
- each channel **must** contain tags:
  - title channel's title
  - description brief text description of channel
  - link an HTML URL to channel's web site
  - language language encoding for channel (e.g., en-us)
  - item from 1 to 15 items
RSS Channel

- each channel may contain additional tags, including:
  - copyright designates content as copyrighted, names holder
  - pubDate date this channel was published
  - lastBuildDate time of last update to channel
  - image some graphic for channel image
- seems like a good idea, in general

Too Good...

- RSS developed by Netscape for a service it wanted to provide: version 0.90
- then UserLand Software did work to simplify original: version 0.91
- further UserLand refinements: 0.92, 0.93, 0.94
- RSS–DEV adopted 0.90 and evolved it into version 1.0
  - based on RDF
- UserLand most recently produced 2.0
- most versions mutually incompatible
Using RSS: source–side

- to provide RSS feed from your site to others:
  - need xml file defining your RSS channel
  - need your httpd server to know about this file
    - so can be served on demand
- need to keep RSS file up–to–date as content
  changes on your site
  - can do by hand
  - many tools to semi– or fully automate

Using RSS: client–side

- RSS–clients
  - receive and render XML of the feed
  - maintain local ‘tracking’ info so can know which
    feeds user subscribes to
    - and (perhaps) also status of feeds
  - may integrate into browser
    - many current browsers will render the xml file as
      plain text; ugly, but readable, sort of
### Sample RSS ‘page’

```xml
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rss version="2.0" xmlns:npr="http://www.npr.org/rss/">
  <channel>
    <title>NPR News: Top Stories</title>
    <link>http://www.npr.org/topics/topic.php?topicId=2&amp;sourceCode=RSS</link>
    <description>NPR News: Top Stories</description>
    <language>en-us</language>
    <copyright>Copyright 2004 National Public Radio</copyright>
    <lastBuildDate>Sun, 28 Nov 2004 18:44:37 EDT</lastBuildDate>
    <image>
      <title>NPR News: Top Stories</title>
      <url>http://www.npr.org/images/npr_news_123x20.gif</url>
      <link>http://www.npr.org/topics/topic.php?topicId=2</link>
    </image>
    <generator>NPR RSS Generator 1.0</generator>
  </channel>
</rss>
```

### Better RSS XML Handling

- **RSS ‘viewer’**
- top picks as recommended by blogspace.com:
  - for Macintosh: NetNewsWire
  - for Windows: SharpReader
  - for Linux: Straw
  - for web: Bloglines
- common to use Python as part of receiving/rendering process
  - a scripting language similar in some ways to Perl and Tcl
Aggregation

- RSS on server does publishing or syndication
- on client–side, collect feeds from multiple sources
  - “...aggregators collect news, weblog and other feeds over the web and aggregate them so the news items are readable from a single place, regardless of their source.”
    -- http://www.nongnu.org/straw
- a site performing aggregation may issue the aggregate as its own RSS feed

Some RSS Feeds...

- tech news: http://slashdot.org/index.rss
- very widespread use with blogs
RSS Standard?

- current RSS users should provide support for 1.0 and 2.0
- what about a single standard?
  - e.g., what’s the IETF say?

RSS Standard?

- what about a single standard?
  - e.g., what’s the IETF say?
- IETF:
  - has “atompub” working group
  - no RFCs yet
  - but has Internet-drafts for proposed \textit{atom} standard
    - The Atom Syndication Format
    - The Atom Publishing Protocol
    - Atom Feed Autodiscovery
Atom

- from “draft-ietf-atompub-format-03."

```xml
<?xml version="1.0" encoding="utf-8"?>
<feed version="draft-ietf-atompub-format-03: do not deploy"
xmlns="http://purl.org/atom/ns#draft-ietf-atompub-format-03">
<head>
<title>Example Feed</title>
<link href="http://example.org/"/>
<updated>2003-12-13T18:30:02Z</updated>
<author>
<name>John Doe</name>
</author>
</head>
<entry>
<title>Atom-Powered Robots Run Amok</title>
<link href="http://example.org/2003/12/13/atom03"/>
<id>venmi://example.org/2003/32397</id>
<updated>2003-12-13T18:30:02Z</updated>
</entry>
</feed>
```

Platform Choices

- implementing solutions requires choosing platforms to perform tasks
  - e.g., web servers, db engines
- many solutions possible
- two main categories:
  - proprietary
  - open-source
What is Open Source?

- licensed software
  - users must accept terms of license in order to use
  - several standard licenses, e.g., GPL
    http://www.gnu.org/licenses/gpl.html
- many developers
  - source code is openly available for anyone to work on
  - organized into code repositories, e.g., SourceForge
    http://sourceforge.net
  - scheduled updates and releases
  - developers distributed world-wide

What is Open Source?

- licensed software
- many developers
- delivered:
  - as source code: user compiles and installs
  - as compiled binaries: user copies and installs
  - as 'packages': user installs (e.g., RPM)
- free
  - no fee is paid to license the software
  - may pay to have 'package' with CDROM and documentation
What’s Available?

- what isn’t?
- operating systems, e.g., Linux
- compilers, e.g., GNU gcc
- databases, e.g., MySQL, SAP
- graphical interfaces, e.g., X window
- network software
  - infrastructure, e.g., web servers, e.g., Apache
  - applications
- apps in all areas

The Bandwagon

- open source software obtained from its creator
  - e.g., Snort from snort.org
- major sites housing open source software
  - e.g., gnu.org
- major vendors
  - e.g., IBM (Linux runs on z90 series mainframes)
  - e.g., HP now offers choice of XP or Linux on new PC purchases
Linux Operating System

- Linux written by Linus Torvalds
  - adapted from minix written by Andrew Tanenbaum
- Derived from UNIX
  - created by Bell Labs in 1970
- Multi-user, multi-tasking OS
- Runs on (nearly) everything
- Current ‘versions’
  - Red Hat Fedora Core
  - Debian, SUSE, Mandrake
  - Major vendors: IBM, HP

Popular Applications

- JBoss: Java-based application server
  http://www.jboss.org
- Tomcat: Java-based web server
  http://jakarta.apache.org/tomcat
- Apache: Web server (written in C)
  http://httpd.apache.org
  - Most widely-used web server on Internet

© http://www.serverwatch.com
Is It Real?

- open source may be OK for a home desktop
- but not suited for ‘real’ work because...
  - not reliable, poor performance
  - it’s not really free (operating costs are higher)
  - no support
  - not fit for mission critical settings
  - not yet mature enough for desktop deployment
  - legal issues over licensing
- do these criticisms hold up?

Price vs Quality

- price not main attraction
  - companies will pay $ for best tech for the job
- linux typically more reliable than NT for server settings
  - case study: Employease:
    - one of company’s NT servers fails each working day
    - at most two linux failures a month, often no failures per month
    - faster: increased capacity 50 to 75 %
    - “...we cannot risk choosing an inferior solution to save money.”
Not Really Free

- may be ‘free’ to get, but operating costs are purportedly higher for training, support, maintenance
- but:
  - no ‘vendor churn’: need newer versions with new licenses ($$)
  - case study: Sabre Group (includes Travelocity) expects to save, over 5 years, “tens of millions”
  - scales: no additional license fees as installation grows
  - “expecting at least 80% reduction in running cost”

No Support

- no customer service rep to yell at
- multiple sources means conflicting answers
- but:
  - world-wide, always-available population of developers
  - major software well documented, often in many languages
  - multiple sources can yield better answers than single vendor (who is sometimes wrong)
  - developers offer training programs
    - e.g., JBoss
No Support

- developers offer service using their software
  - e.g., Sourcefire (creators of Snort)

Not For Mission Critical

- risky in terms of security, reliability, maintenance
- case study: Banca Popolare di Milano:
  - had legacy system 90 million lines of COBOL
  - used legacy integration tool (Jacada) connecting COBOL code to IBM’s WebSphere in linux partition
Maturity

- still just for computer nerds
- Siemens Business Services study: linux suitable for desktop use by “nontechnical” workers
- case study: Baylis Distribution
  - moved major service to linux-based platform
  - then moved desktop users to linux environment
  - users see same ‘desktop’ no matter what PC they log on to; no personalized environment problems
  - “We’ve got better control, better upgradability, and better traceability” at “around half” the $

Legalese

- concern over licensing: who owns rights to code, what affect on right to use?
- SCO suing IBM over linux code purportedly proprietary UNIX code now owned by SCO
  - SCO’s claim to the linux code not established
  - almost no one expects the suit (or related suits) to have any impact on linux users
  - some sources provide indemnification to users
    - e.g., JBoss Group, HP, Red Hat and Novell, Sun
Grid Computing

- make a collection of distributed resources appear to a user as a single, coherent, computing resource
  - e.g., cpu, storage, visualization & display
  - even if different platforms are involved
- connect resources over a network to make available to user
- may dynamically create such collections
- start with a research-oriented system...

The TeraGrid Project

- NSF stimulated project to create resource for researchers
- consists of:
  - 20 teraflops over 5 distinct locations
  - storage management of 1 Pb (2^{50} bytes)
  - tightly integrated components in each cluster
  - connected to other clusters via 40 Gbs network
- e.g., linux cluster (Itanium® processors): 1 Tf
TeraGrid

Grid Computing for Mortals

- same goal: creating computing resource from separate distributed resources
- use standard Internet
- need special software to build grid
  - currently emerging std is Globus Toolkit (2.2)
  - “Legion” from Avaki (Grimshaw at U. Virginia)
Grid Software

- need to ‘virtualize’ resources
- Open Grid Services Architecture (OGSA)
  - common ‘view’ of a resource and how to use it
- for service ensembles, OGSA supports:
  - creation
  - maintenance
  - application
- see http://www.ogsa.org/ogsa
The Emerging Grid

- grid: "...emergence of a new infrastructure upon which first science, and then the whole economy, will be built" (Larry Smarr)
- do for computing what Internet did for documents
- aiming at worldwide governance & standards body analogous to IETF: http://www.gridforum.org

IPSEC

- works at IP layer
- but is connection-oriented
- each connection is a Security Association (SA)
  - simplex connection
  - need 2 SAs for two-way communication
- 2 modes:
  1. transport
  2. tunnel
- encryption “always on”
  - but IPsec doesn’t dictate particular algorithms
  - can use “null” algorithm for no encryption (RFC 2410)
Security Associations

- ‘secure’ connection from sender to receiver
- parameters negotiated at set-up
  - e.g., key
- database entry at receiver has parms for each SA
- SAs uniquely identified by:
  1. Security Parameters Index (SPI)
  2. IP destination address
  3. security protocol identifier

SA characteristics

- sequence number: 32-bit counter
- sequence counter o’flow: does counter overflow trigger “auditable” event and lock-out this SA
- anti-replay window: range of allowed (expected) sequence numbers
- lifetime: time interval or byte-count after which SA is no longer valid
- IPSec protocol mode: transport or tunnel
- Path MTU: max size that doesn’t need fragmenting
- Hdr specific parms: see header descriptions
Components of IPsec

3 main components:
2. AH: Authentication Header
   - integrity: can’t make undetected changes to msg
   - end receiver has ability to authenticate
     - eliminate spoof attacks
     - guard against replay attacks
3. ESP: Encapsulating Security Payload
   - confidentiality of msg contents
   - some traffic flow confidentiality
4. key management mechanism

AH Format

<table>
<thead>
<tr>
<th>0</th>
<th>8</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>nexheader</td>
<td>payload length</td>
<td>reserved</td>
<td></td>
</tr>
<tr>
<td>security parameters index (SPI)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sequence number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>authentication data (variable)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ESP Packet Format

- security parameters index (SPI)
- sequence number
- payload data (variable)
- padding (0 – 255 bytes)
- padding (0 – 255 bytes)
- pad length
- nextheder
- authentication data (variable)

AH & ESP Operation

- both operate in
  - transport mode
  - tunnel mode
- IPsec allows one of AH or ESP but not both at a time
Transport Mode

- typically for end-to-end use
- payload in packet is protected
  - in IPv4, payload immediately follows IP hdr
- ESP encrypts (+ optionally authenticates) IP payload but not header
- AH authenticates IP payload + selected parts of IP header

Tunnel Mode

- protects entire IP packet
- add AH or ESP fields to IP packet then re-wrap in new IP packet
  - can have different src/dst address (e.g., firewall to firewall)
  - “inner” packet is “tunneled” through network
  - NAT issues
- ESP encrypts (+ optionally authenticates) entire inner IP packet
- AH authenticates entire inner packet + selected parts of outer IP header
ESP in Transport & Tunnel Modes

IPv4 example

Transport:

<table>
<thead>
<tr>
<th>IP hdr</th>
<th>ESP header</th>
<th>TCP</th>
<th>data</th>
</tr>
</thead>
</table>

Tunnel:

<table>
<thead>
<tr>
<th>IP hdr</th>
<th>ESP header</th>
<th>IP hdr</th>
<th>TCP</th>
<th>data</th>
<th>ESP trailer</th>
<th>ESP authent.</th>
</tr>
</thead>
</table>

encrypted

authenticated

Fortification

- installations want to
  - protect assets from damage or theft originating outside
  - protect assets from damage or theft originating inside
- introduce model with **single access path** to outside: the firewall

Internet (outside world)
Firewall Components

- **packet filter**: standard router + packet inspection function
- packet inspection is rule-based
- rules allow, e.g.,
  - allow/deny all packets to a particular port
  - allow/deny all packets to/from a particular IP address
  - default rules for packets not covered by current rules

Firewall Components

- **application gateway**: examines messages themselves
  - e.g., mail gateway examines mail messages and decides whether to forward or drop message based on
    - size of message
    - headers
    - perhaps even content
**Generic Firewall**

- typically, a firewall looks like:

  ![Diagram of a generic firewall]

**Firewalls: Not a Complete Solution**

- things a firewall cannot defend against:
  - bad guy uses false source addresses in datagrams
  - can't handle encrypted messages
  - won’t catch ‘hidden’ messages
  - attacks from inside the firewall
  - and...
Creating a Connection

- a TCP connection is requested (active open) by host sending a SYN to destination
- dest responds with its own SYN and a piggy-backed ACK
  - \( \Rightarrow \) dest allocated some internal resources to support the connection
  - these resources stay ‘reserved’ for a short time and are then released if connection is unused

Creating Lots of Connections

- what happens if bad guy sends thousands of SYN requests to a site? millions?
- server runs out of resources to support legitimate requests so legitimate users are denied access to the service
- \( \Rightarrow \) denial of service (DOS) attack
- countermeasure: packets coming from one source address are filtered to the “bit-bucket”
  - still takes some time, but not as resource crippling
Creating Lots of Connections

- counter-countermeasure: don’t initiate attack from only one place
  - first, break into a number of nodes scattered over the network
  - arrange for each of those nodes to begin a DOS at a certain time
  - ⇒ distributed denial of service (DDOS) attack
- harder to identify
  - can look like increased interest in server

Private Network

- to protect networked assets, organizations use private networks
- can be real: lease dedicated lines between routers at sites
  - but leased lines expensive
Private Network

- to protect networked assets, organizations use private networks
- can be real
- can be virtual (VPN):
  - run over public network, often Internet itself
  - combine several elements we've seen to achieve effect of dedicated private network over public network

Virtual Private Network

use:
- **tunnels**: carry encrypted traffic between
- **firewalls** as single-access points
- use IPSEC's ESP for tunneling then have single SA for each 'channel': confidentiality + integrity
What's Out There?

- why this concern for keeping ‘barbarians at the gate’?
- “malware” is out there...

Taxonomy of Malware

- malicious software
  - needs host program
    - trap door
    - logic bomb
    - Trojan horse
  - independent
    - virus
    - worm
    - zombie
Taxonomy of Malware

**trap doors**:
- built into applications by programmers to give them access without going through usual authentication and security (the front door)
- accessed through special input sequence, or some unusual combination of events
- can serve as access point to assets or to activate other threats, e.g., Trojan horse

```
trap door  logic bomb  Trojan horse
           virus
           worm
           zombie
```
**Taxonomy of Malware**

**Trojan horse:**
- a useful appearing program that contains other code which, when invoked, performs undesirable actions
- usually embedded in innocent appearing program, so can gain access to a system
- once installed, hidden code can run to perform actions favourable to an intruder (e.g., use user’s abilities to do actions that let intruder gain greater access)

**Taxonomy of Malware**

**Worm:**
- a program whose goal is to spread itself across many hosts on a network (perhaps also doing mischief at each host)
- may use only network services to locate further targets, or use applications (e.g., email)
Taxonomy of Malware

**zombie:**
- a program that covertly ‘takes over’ a network reachable computer, installs software on it, and uses it to launch attacks on third parties that cannot now be easily traced to true attacker
- popular for denial of service attacks

malicious software

*trap door*  *logic bomb*  *Trojan horse*

**Viruses**

- in nature:
Viruses

- a (usually) very tiny program that “attaches” itself to another program
- when the carrier program runs, the virus runs
- virus can do anything the privilege level of the user allows it to
  - delete files
  - send email
- virus spreads by attaching itself to as many other programs as it can, and/or copying itself to special locations
  - e.g., system areas on disk

Viruses

- a trigger event occurs at which time other virus functions may begin executing
  - e.g., drawing pictures on screen
  - e.g., deleting files
- common triggers:
  - “magic” dates (e.g., Michaelangelo virus)
  - number of copies this copy has made
Categories of Virus

- parasitic: commonest type
  - attached to an executable, runs when executable runs and replicates

Categories of Virus

- parasitic
- memory-resident:
  - becomes memory resident and infects every program run thereafter
Categories of Virus

- parasitic
- memory-resident
- boot-sector:
  - infects specific boot-sector area of disk, spreading when system is booted from that infected boot sector

Categories of Virus

- parasitic
- memory-resident
- boot-sector
- stealth:
  - uses stealth tricks to avoid detection
  - e.g., compression
Categories of Virus

- parasitic
- memory-resident
- boot-sector
- stealth
- polymorphic:
  - mutates with each copy to avoid presenting recognizable viral signature to anti-virus checkers

Categories of Virus

- parasitic
- memory-resident
- boot-sector
- stealth
- polymorphic
- macro:
  - not machine executable, but application macro ‘executable’
  - carried not in programs but in documents
  - targets apps like Word, Excel
Virus Countermeasures

- cleanliness: keep viruses away from system
  - use only sealed known–original applications from trusted vendors
    - not infallible
- once infected:
  - detect – identify – eradicate
  - detection involves software that can detect presence of virus in executable image
  - identification: virus has particular signature
  - eradication involves removal of virus code from infected system, and those it may have infected