Professional Masters in Information and Cyber Security (PMICS)

Computer Science and Engineering,
University of Dhaka

CSE 802

Information Security Fundamentals

DNS

- All internet communication protocols require numerical addresses.
- Numerical addresses are much too cumbersome for humans to keep track of.
- When you ask your computer to make a connection with some remote machine, you are likely to specify a symbolic hostname for that machine.
- TCP/IP cannot send a single packet to the destination without knowing the numerical address.
- It is infeasible for a computer to store the symbolic hostname to numerical IP address mappings.
- Considering that the internet is constantly expanding, it is impossible to keep such a central repository updated on a second-by-second basis

DNS

- Stands for domain name service/domain name server/domain name system/domain name space.
- Translates hostnames into numerical IP addresses and vice versa.
- It lists email exchange server accepting email from different domains.
  - Domain name to email exchange server mapping --→MX record in DNS
- Hostname and IP address does not match on a one-to-one basis.
  - Many hostnames may correspond to an IP address (example virtual hosting)
  - A single hostname may corresponds to many IP addresses (for fault tolerance and load distribution)

DNS

- the largest and most important distributed database.
- It is an open and openly extendible database.

- DNS Hijacking on Non-Existent Domain Names

your browser makes a request to the ISP DNS server for the IP address associated with a hostname that does not exist (because you made a spelling error in the URL), the DNS server is supposed to send back the NXDOMAIN error message to your browser. Instead, the ISP's DNS server sends back a browser redirect to an advertisement-loaded website that the ISP wants you to look at. Or, the ISP's DNS server may send you suggestions for domains that are similar to what your browser is looking for.

Linux DNS Files

- /etc/hosts in ubuntu provided mapping between IP addresses and hostnames when a system started before DNS can be referenced.
- In the absence of a name server, a network program accesses this file for mapping.

Following is a sample `/etc/hosts` file:

```
IPAddress     Hostname                Alias
127.0.0.1                   localhost               deep.openna.com
208.164.186.1               deep.openna.com         deep
208.164.186.2               mail.openna.com         mail
208.164.186.3               web.openna.com          web
```

Linux DNS Files

- /etc/host.conf --- tells what order the name resolver should search for hostnames-to-IP addresses mapping.
- cat /etc/host.conf

  order hosts, bind
- /etc/resolv.conf --- lists the nameservers (either using IP addresses or with symbolic hostnames) to use by the name resolver programs.
- The entries in this file are automatically generated by the networking software in a computer and these entries change when moving from one location to another in a different networking domain.

Linux DNS Files

- if you change any of the network config files, such as, say, /etc/hosts, you would need to restart the network service by

  sudo /etc/init.d/network restart

  or, by

  sudo service network-manager restart

DNS Name Resolver

- A library consist of several functions to perform several tasks:
  --- gethostbyname()
  translate the symbolic hostnames into IP address.
  --- gethostbyaddr()
  translate IP address into hostname
  --- getaddrinfo()
  translate the symbolic hostnames into IP address.
  --- getnameinfo()
  translate IP address into hostname.

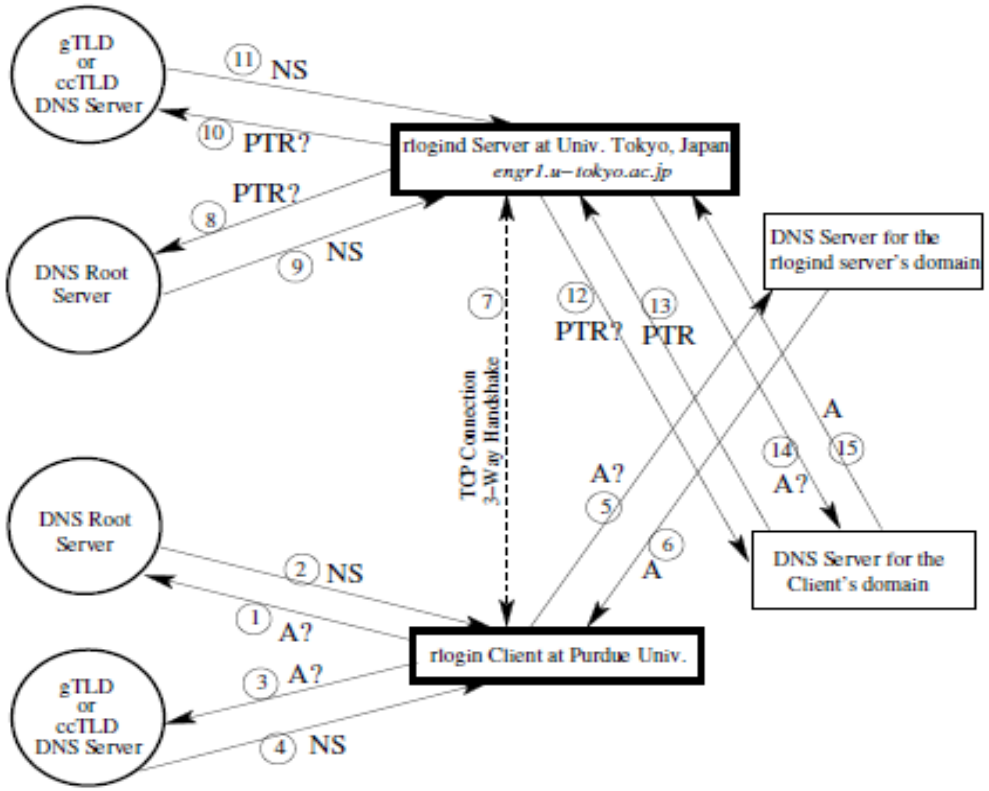
An Example of DNS Hijacking

You may download malware by clicking on a URL in a spam email which may overwrite the entries in the file /etc/resolv.conf. This would cause your name resolution requests to be serviced by a rogue DNS. When that happens, your browser may end up visiting a malicious website that is made to look like the one you were actually trying to reach. If you fall prey to such a subterfuge, you could end up giving your personal information, such as your bank account information, to a bunch of bad guys.

Structure of Worldwide DNS

- At the top of the hierarchy are the 13 root servers.
- Can be found at */usr/share/dns/root.hints* file.
- There are 13 root DNS servers with names like a.root-servers.net, b.root-servers.net, c.root-servers.net, etc.
- only 6 of them have fixed geographical locations, all in the US. Seven others are replicated at a large number of locations all around the world.
- When a host sends a query for name resolution to one of the 13 root servers, the root server responds back with the IP address of either a Generic Top Level Domain (gTLD) DNS server or IP address of a Country Code Top Level Domain (ccTLD) DNS server.
- If a root server receives a query for, say, the '.com' domain, the root server sends back the IP address of one or more gTLD nameservers in charge of the '.com' domain. On the other hand, if a root server receives a query for, say, the '.jp' domain, the response back from the root consists of the IP address of the ccTLD server in charge of the '.jp' domain.
- The gTLD have specific names, fixed IP addresses, and fixed physical locations,
- The ccTLD may have any name, any arbitrary IP address that is registered with any ISP and any physical location.

Illustration of Extensive DNS Lookups for a Simple Client-Server Interactions



Command executed at the rlogin client at Purdue: rlogin engr1.u–tokyo.ac.jp –l joe

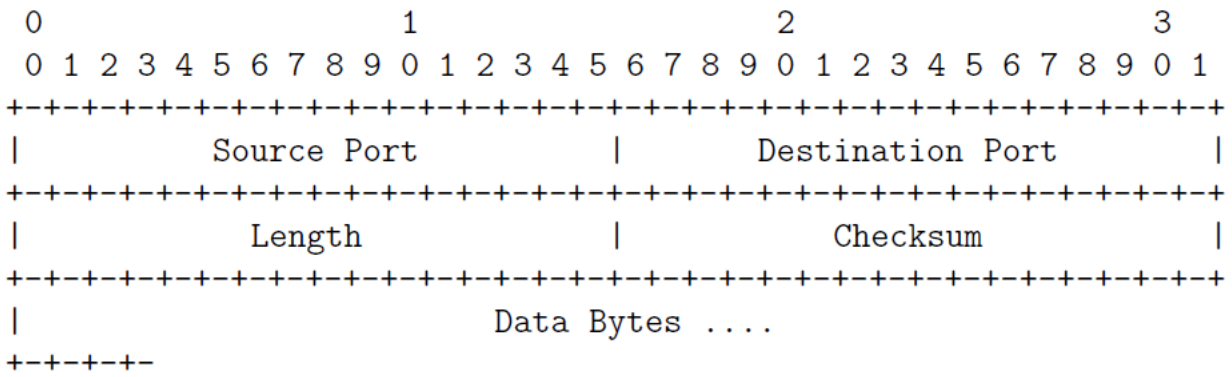
A?	Query to a nameserver for an IPv4 address
A	Resource record returned by nameserver with an IPv4 address
PTR?	A pointer query to a nameserver (for the hostname associated with an IPv4 address)
PTR	Pointer record returned by a nameserver for a pointer query (This would be the hostname)
NS	A Name Server record returned by a root DNS server

A client makes remote connection with a machine in University of Tokyo using

rlogin remote_machine_hostname -l your_name

DNS Request for Name Lookup

- a DNS request for name lookup is sent out in the form of a UDP packet.



DNS Name Lookup using TCPDUMP

```
ssh  engr.u-tokyo.ac.uk
```

PACKET 1 (from my laptop to a root nameserver):

```
10:23:23.205572 IP (tos 0x0, ttl 64, id 45217, offset 0, flags [none], proto UDP (17), length 75)

192.168.1.105.22579 > 198.41.0.4.53: [udp sum ok] 47551 [1au] A? engr.u-tokyo.ac.uk. ar: . OPT UDPsize=4096 OK (47)
```

PACKET 2 (from the root nameserver to my laptop):

```
10:23:23.279603 IP (tos 0x20, ttl 52, id 19828, offset 0, flags [none], proto UDP (17), length 720)

198.41.0.4.53 > 192.168.1.105.22579: [udp sum ok] 47551- q: A? engr.u-tokyo.ac.uk. 0/13/15 ns:
uk. [2d] NS ns4.nic.uk., uk. [2d] NS ns1.nic.uk., uk. [2d] NS nsd.nic.uk., uk. [2d] NS ns2.nic.uk.,
uk. [2d] NS ns3.nic.uk., uk. [2d] NS ns7.nic.uk., uk. [2d] NS ns5.nic.uk., uk. [2d] NS nsa.nic.uk.,
uk. [2d] NS ns6.nic.uk., uk. [2d] NS nsb.nic.uk., uk. [2d] NS nsc.nic.uk., uk. [1d] NSEC,
uk. [1d] RRSIG ar:
ns1.nic.uk. [2d] A 195.66.240.130, ns1.nic.uk. [2d] AAAA 2a01:40:1001:35::2, ns2.nic.uk. [2d] A 217.79.164.131,
ns3.nic.uk. [2d] A 213.219.13.131, ns4.nic.uk. [2d] A 194.83.244.131, ns4.nic.uk. [2d] AAAA 2001:630:181:35::83,
ns5.nic.uk. [2d] A 213.246.167.131, ns6.nic.uk. [2d] A 213.248.254.130, ns7.nic.uk. [2d] A 212.121.40.130,
nsa.nic.uk. [2d] A 156.154.100.3, nsa.nic.uk. [2d] AAAA 2001:502:ad09::3, nsb.nic.uk. [2d] A 156.154.101.3,
nsc.nic.uk. [2d] A 156.154.102.3, nsd.nic.uk. [2d] A 156.154.103.3, . OPT UDPsize=4096 OK (692)
```

DNS Name Lookup using TCPDUMP

PACKET 3 (from my laptop to a nameserver for the uk domain):

```
10:23:23.283030 IP (tos 0x0, ttl 64, id 39865, offset 0, flags [none], proto UDP (17), length 75)

192.168.1.105.46921 > 195.66.240.130.53: [udp sum ok] 27013 [1au] A? engr.u-tokyo.ac.uk. ar: . OPT UDPsize=4096 OK (47)
```

PACKET 4 (from the nameserver for uk domain to my laptop):

```
10:23:23.407573 IP (tos 0x20, ttl 52, id 38716, offset 0, flags [none], proto UDP (17), length 711)

195.66.240.130.53 > 192.168.1.105.46921: [udp sum ok] 27013- q: A? engr.u-tokyo.ac.uk. 0/11/1 ns:
ac.uk. [2d] NS ns0.ja.net., ac.uk. [2d] NS ws-fra1.win-ip.dfn.de., ac.uk. [2d] NS ns2.ja.net.,
ac.uk. [2d] NS ns4.ja.net., ac.uk. [2d] NS sunic.sunet.se., ac.uk. [2d] NS ns3.ja.net.,
ac.uk. [2d] NS ns.uu.net.,
u1fmklfv3rdcnamdc64sekgcdp05bbiu.uk. [2d] Type50, u1fmklfv3rdcnamdc64sekgcdp05bbiu.uk. [2d]
RRSIG, ptc0fm5i0qano6f75ivbss4dg368caci.uk. [2d] Type50, ptc0fm5i0qano6f75ivbss4dg368caci.uk.
[2d] RRSIG ar: . OPT UDPsize=4096 OK (683)
```

DNS Name Lookup using TCPDUMP

PACKET 5 (from my laptop to a gTLD nameserver for the IP address of `ns.uu.net` mentioned in the reply in Packet 4):

```
10:23:23.411002 IP (tos 0x0, ttl 64, id 60810, offset 0, flags [none], proto UDP (17), length 66)

192.168.1.105.36824 > 192.55.83.30.53: [udp sum ok] 56478% [1au] A? ns.uu.net. ar: . OPT UDPsize=4096 OK (38)
```

PACKET 6 (from my laptop to another gTLD nameserver for the IP address of `ns.uu.net` mentioned in the reply in Packet 4):

```
10:23:23.411384 IP (tos 0x0, ttl 64, id 53824, offset 0, flags [none], proto UDP (17), length 66)

192.168.1.105.37664 > 192.54.112.30.53: [udp sum ok] 62789% [1au] AAAA? ns.uu.net. ar: . OPT UDPsize=4096 OK (38)
```

The Domain Name System

- Internet is divided into a tree of zones.
- Each zone, consisting of a Domain Name Space served by a DNS nameserver that, in general, consists of two parts:
  - an Authoritative Nameserver for the zone which directly knows the hostname-to-IP address mappings (from zone file) ; and
  - a Recursive Nameserver for all other IP addresses.

Fully Qualified Domain Name (FQDN):

- The root domain is represented by a period
- Example: ecn.purdue.edu.
- Read from the right to left.

dig Utility

- Stands for **Domain Information Grooper**.
- Used to interrogate DNS nameservers for information about the host IP addresses, mail exchanges, nameservers for other domains
- Included in library dnsutils (ubuntu).
- Example: dig moonshine.ecn.purdue.edu

dig Utility

```
; <<>> DiG 9.4.1-P1 <<>> moonshine.ecn.purdue.edu
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50449
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 6, ADDITIONAL: 2

;; QUESTION SECTION:
;moonshine.ecn.purdue.edu.  IN A

;; ANSWER SECTION:
moonshine.ecn.purdue.edu.  86377 IN A 128.46.144.123


;; AUTHORITY SECTION:
ecn.purdue.edu.  81544 IN NS ns1.rice.edu.
ecn.purdue.edu.  81544 IN NS ns2.purdue.edu.
ecn.purdue.edu.  81544 IN NS harbor.ecn.purdue.edu.
ecn.purdue.edu.  81544 IN NS ns2.rice.edu.
ecn.purdue.edu.  81544 IN NS pendragon.cs.purdue.edu.
ecn.purdue.edu.  81544 IN NS ns.purdue.edu.

;; ADDITIONAL SECTION:
ns2.rice.edu.  3550 IN A 128.42.178.32
ns2.purdue.edu.  81544 IN A 128.210.11.57

;; Query time: 1 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sat Mar 29 11:13:37 2008
;; MSG SIZE  rcvd: 214
```

Resource Record (RR):

1. A fully qualified domain name (FQDN)

2. Time-to-live (TTL)

3. The class of the record, such as IN

4. The type of the record.
   A: for IPv4 address record
   AAAA: for IPv6 address record
   NS: for a nameserver record
   PTR: for pointer record
   MX: for a mail exchange server for a given host.

5. Record data

dig Utility

```
; <<>> DiG 9.4.1-P1 <<>> -x 58.9.62.229
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61596
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

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

;; ANSWER SECTION:
229.62.9.58.in-addr.arpa. 604560 IN PTR ppp-58-9-62-229.revip2.asianet.co.th.

;; AUTHORITY SECTION:
9.58.in-addr.arpa. 604560 IN NS conductor.asianet.co.th.
9.58.in-addr.arpa. 604560 IN NS piano.asianet.co.th.
9.58.in-addr.arpa. 604560 IN NS clarinet.asianet.co.th.

;; ADDITIONAL SECTION:
piano.asianet.co.th. 86160 IN A 203.144.255.71
conductor.asianet.co.th. 86160 IN A 203.144.255.72
clarinet.asianet.co.th. 86160 IN A 203.144.225.242

;; Query time: 1 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sat Mar 29 15:20:28 2008
;; MSG SIZE  rcvd: 207
```

```
dig -x 58.9.62.229
```

dig Utility

```
; <<>> DiG 9.9.3-rpz2+rl.13214.22-P2-Ubuntu-1:9.9.3.dfsg.P2-4.....
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44572
;; flags: qr rd ra; QUERY: 1, ANSWER: 7, AUTHORITY: 0, ADDITIONAL: 15

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4000
;; QUESTION SECTION:
;nyt.com.                       IN      MX

;; ANSWER SECTION:
nyt.com.                300     IN      MX      30 ASPMX4.GOOGLEMAIL.com.
nyt.com.                300     IN      MX      10 ASPMX.L.GOOGLE.com.
nyt.com.                300     IN      MX      20 ALT1.ASPMX.L.GOOGLE.com.
nyt.com.                300     IN      MX      30 ASPMX3.GOOGLEMAIL.com.
nyt.com.                300     IN      MX      20 ALT2.ASPMX.L.GOOGLE.com.
nyt.com.                300     IN      MX      30 ASPMX5.GOOGLEMAIL.com.
nyt.com.                300     IN      MX      30 ASPMX2.GOOGLEMAIL.com.

;; ADDITIONAL SECTION:
ASPMX.L.GOOGLE.com.             115     IN      A       74.125.142.26
ASPMX.L.GOOGLE.com.             185     IN      AAAA    2607:f8b0:4001:c03::1b
ALT1.ASPMX.L.GOOGLE.com.        139     IN      A       74.125.29.26
ALT1.ASPMX.L.GOOGLE.com.        130     IN      AAAA    2607:f8b0:400d:c04::1a
ASPMX3.GOOGLEMAIL.com.          128     IN      A       74.125.131.27
ASPMX3.GOOGLEMAIL.com.          275     IN      AAAA    2607:f8b0:400c:c03::1a
ALT2.ASPMX.L.GOOGLE.com.        289     IN      A       74.125.131.26
ALT2.ASPMX.L.GOOGLE.com.        240     IN      AAAA    2607:f8b0:400c:c03::1a
ASPMX5.GOOGLEMAIL.com.          184     IN      A       173.194.65.27
ASPMX5.GOOGLEMAIL.com.          106     IN      AAAA    2a00:1450:4013:c00::1b
ASPMX2.GOOGLEMAIL.com.          195     IN      A       74.125.29.26
ASPMX2.GOOGLEMAIL.com.          172     IN      AAAA    2607:f8b0:400d:c04::1a
ASPMX4.GOOGLEMAIL.com.          103     IN      A       173.194.78.26
ASPMX4.GOOGLEMAIL.com.          33      IN      AAAA    2a00:1450:400c:c00::1a

;; Query time: 50 msec
;; SERVER: 127.0.1.1#53(127.0.1.1)
;; WHEN: Tue Mar 25 22:16:22 EDT 2014
;; MSG SIZE  rcvd: 520
```

- dig nyt.com MX

dig Utility

```
dig  @ns1.rice.edu  +nocmd  moonshine.ecn.purdue.edu
```

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 33037
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;moonshine.ecn.purdue.edu.  IN A

;; ANSWER SECTION:
moonshine.ecn.purdue.edu.  86400 IN A 128.46.144.123

;; Query time: 86 msec
;; SERVER: 128.42.209.32#53(128.42.209.32)
;; WHEN: Sun Mar 30 11:22:27 2008
;; MSG SIZE  rcvd: 58
```

dig Working Procedure

- examines /etc/resolv.conf file for the nameservers to send the query.
- Content of /etc/resolv.conf

```
nameserver    127.0.0.1
nameserver    68.87.72.130
nameserver    68.87.77.130
```

host and nslookup Utilites

```
host  moonshine.ecn.purdue.edu
```

returns

```
moonshine.ecn.purdue.edu  has  address  128.46.144.123
moonshine.ecn.purdue.edu  mail  is  handled  by  10 mx.ecn.purdue.edu.
```

and

```
nslookup  moonshine.ecn.purdue.edu
```

returns

```
Server:         127.0.0.1
Address:        127.0.0.1#53

Non-authoritative answer:
Name:   moonshine.ecn.purdue.edu
Address: 128.46.144.123
```

```
nslookup  moonshine.ecn.purdue.edu  ns2.rice.edu
```

which returns

```
Server:ns2.rice.edu
Address:128.42.178.32#53

Name: moonshine.ecn.purdue.edu
Address: 128.46.144.123
```

```
nslookup  -type=NS  moonshine.ecn.purdue.edu
```

which returns

```
Server:127.0.0.1
Address:127.0.0.1#53

Non-authoritative answer:
*** Can't find moonshine.ecn.purdue.edu: No answer

Authoritative answers can be found from:
ecn.purdue.edu
origin = harbor.ecn.purdue.edu
mail addr = hostmaster.ecn.purdue.edu
serial = 2009040816
refresh = 10800
retry = 3600
expire = 3600000
minimum = 86400
```

DNS Cache

- Sending every DNS lookup request to the root servers places a great deal of burden on the root servers slowing down the name lookup process.

- you are within the purdue.edu domain and you point your browser to www.nyt.com. The browser will send that URL to one of the nameservers of the purdue.edu domain. If this is the first request for this URL received by the nameserver for purdue.edu, the nameserver will forward the request to the nameserver for the 'com' domain, and the name lookup will proceed in usual manner. If this was not the first request for the name resolution of www.nyt.com, it is likely that the local nameserver would be able to resolve the URL by looking into its own cache.

- In general, various client applications such as mail clients, web browsers, etc. maintain their own DNS caches usually with very short caching times (typically 1 minute but which can be as long as 30 minutes).

- The operating system may carry out some local name resolution by looking up the information in /etc/hosts for any direct hostname-to-IP address mappings.

- The operating system may also maintain a local cache for the previously resolved hostnames with relatively short caching times (of the order of 30 minutes) for the information stored.

Time to Live (TTL)

- An authoritative DNS server sends back a time interval known as the TTL (Time to Live) in addition to the IP address.
- The TTL specifies the time interval for which the response can be expected to remain valid. DNS cache stores both an IP address and its TTL.
- For all DNS queries for the same hostname made within the TTL window, the local name-resolver working with the DNS server will return the cached entry.
- The TTL value associated with a hostname is set by the administrator of the authoritative DNS server.
- While DNS caching makes the hostname resolution faster, any changes to the DNS do not always take effect immediately and globally.

Time to Live (TTL)

- An authoritative nameserver is known as a publishing nameserver and a recursive nameserver as a caching nameserver.
- Google runs the world's largest recursive name server that handles around 400 billion name requests a day. There are two IPv4 and two IPv6 addresses associated with this name server. The IPv4 addresses are 8.8.8.8 and 8.8.8.4 and the IPv6 addresses are 2001:4860.4860::8888 and 2001:4860.4860::8884.
- By forcing the downstream recursive DNS servers to fetch the IP bindings associated with a given name more often, it is possible to evenly distribute the incoming load targeting a particular symbolic hostname.
- The TTL associated with all such top-level gTLDs and the ccTLDs domain servers is 172800 seconds (48 hours).

Time to Live (TTL)

- **if root servers were taken down by an adversary, the information about the TLD would reside in the lower-level nodes of the DNS tree roughly two days giving long enough time for remedial action.**
- **If an adversary took down the gTLDs and the ccTLDs — probably an impossible feat because many of the gTLDs are geographically replicated and because of the ccTLDs are much more numerous the slave servers for those TLDs would provide immediate relief.**

BIND

- BIND (Berkeley Internet Name Daemon) is the most commonly used implementation of a domain name server (DNS).

- The BIND software package consists of the following three components
  - a DNS server (the server program itself is called named in the Ubuntu install of BIND)
  - a DNS name resolver library (the software package that queries DNS servers for information such as the IP address for a given symbolic host name)
  - tools such as dig, host, nslookup, etc., for verifying the proper operation of the DNS server

BIND

- The named server daemon listens on port 53 for both UDP and TCP requests. Most commonly the incoming name queries will use the UDP transport and the answer returned by the nameserver will also be a UDP message.
- However, if the response to be returned to a client is longer than 1024 bytes, the nameserver will switch to the TCP protocol on the same port.

Running a Process in Chroot Jail

- when you run an executable on a Linux machine, it runs with the permissions of the user that started up the executable.

- Suppose, a web server is fired up by a sysadmin as root. Unless some care is taken, the child processes spawned by the web server will have root access. All of the server's interaction with the machine on which it is running would be as root. A web server must be able to write to local files and to also execute them.

- when the main HTTPD process starts up as being owned by root, it may spawn child processes as 'nobody'

- *chroot* command allows the sysadmin to force the program to run in a specified directory and without allowing access from that directory to any other part of the file system.

Running a Process in Chroot Jail

- You can invoke HTTPD as `chroot /www httpd`

- All pathnames to any resources called upon by HTTPD would now be with respect to the node /www. The node /www now becomes the new '/' for the httpd executable. Anything not under /www will not be accessible to HTTPD.

DNS Cache Poisoning

- Entering in the cache a fake IP address for a hostname, a domain name, or another nameserver.
- The use of a 16-bit Transaction ID integer sent with every DNS query makes DNS cache poisoning difficult. This integer is supposed to be randomly generated.
- When an application running on your computer needs to resolve a symbolic hostname for a remote host, it sends out a DNS query along with the 16-bit Transaction ID integer.
- If the nameserver to which the DNS query is sent does not contain the IP address either in its cache or in its zones for which it has authority, it will forward the query to another nameservers. Each such query will be accompanied with its own 16-bit Transaction ID number.

DNS Cache Poisoning

- When a nameserver is able to respond to a DNS query with the IP address, it returns the answer along with the Transaction ID number so that the recipient can identify the corresponding query.

- As long as the TCP or UDP port number, the IP address and the Transaction ID from the remote host are correct, the reply to the query is considered to be legitimate.

DNS Cache Poisoning Procedure

1. Suppose an attacker wants to poison the cache of the nameserver running on the machine *harbor.ecn.purdue.edu* by placing in its cache an incorrect IP address for *amazon.com* domain.
2. The attacker starts by asking the DNS server at *harbor.ecn.purdue.edu* to carry out the name

   dig amazon.com @harbor.ecn.purdue.edu

3. Assume that there was no recent name lookup for *amazon.com* at the DNS server, it will make an NS query to the nameserver of the com top-level domain for the IP addresses of the nameservers in charge of the amazon.com domain.

4. This NS query issued by the nameserver at *harbor.ecn.purdue.edu* will contain a pseudorandom Transaction ID integer.

DNS Cache Poisoning Procedure

1. In another window the attackers will simultaneously run a script that floods ***harbor.ecn.purdue.edu*** with manually crafted DNS reply packets containing the wrong IP address.
2. Each reply contains a different Transaction ID integer, with the hope that the Transaction ID in one of those fake replies will match the Transaction ID in the query sent out by harbor.
3. There is now a race between the correct reply from the nameserver that has the legitimate IP address for the ***amazon.com*** domain and the flood of fake replies sent by the attacker. If the Transaction ID integers used by the DNS server are sufficiently predictable, the attacker could get lucky.
4. The DNS server running at harbor accepts the first reply that *looks* legitimate (in the sense that it contains the correct Transaction ID number).

DNS Cache Poisoning Procedure

1. A fake reply is allowed to contain information in its Additional Section, information that was not specifically requested in the queries emanating from harbor. It would be stored away by the DNS server on harbor as accepted.
2. The attacker can include a wrong IP address for the nameservers assigned to the ***amazon.com*** domain.

   pdns1.ultradns.net. 86400 IN A xxx.xxx.xxx.xxx

   where xxx.xxx.xxx.xxx stands for the wrong IP address. In this manner, you could also hijack the nameservers for the ***amazon.com*** domain.

3. Additional field cannot be restricted as it is required to remove DNS traffic burden.

4. The attacker can associate the longest possible TTL with the fake replies.

DNS Cache Poisoning Procedure

- Success with a DNS cache poisoning attack depends on how deep an understanding the attacker has of the pseudorandom number generator used by the attacked nameserver.

- Earlier versions of BIND did not randomize the Transaction IDs; the numbers used were purely sequential.

- BIND's implementation of the DNS protocol actually sends multiple simultaneous queries for the same symbolic name, each with a different Transaction ID number.

- If the attacker somehow gains knowledge of the previously used Transaction IDs by the attacked nameserver, he/she may be able to predict with a high probability the next Transaction ID that the attacked nameserver will use.

Email Basics

Source: https://www.youtube.com/watch?v=m8TLN--aZic

Spam Email

- Spam is a major source of malware that infects computers
- It tries to lure you into clicking on URLs of websites that serve as hosts for viruses, worms, and trojans.

- There is also another kind of spam emails generated by legitimate businesses and organizations that you either have no interest in reading or have no time for following up on.

- Spam filter can be two types:
  - Statistical filter/ Bayesian filter:
    A statistical filter with sufficiently low “falses” requires too many samples of a certain type of spam before blocking such messages in the future.
  - Regular Expression based filter:
    With a regular-expression based filter, once you see a spam message that has leaked through, it is not that difficult to figure out variations on that message that the spammers may use in the future. You can design a short regular expression to block spam emails.

Reading Emails in Unix/Linux Machine

- Reading email using web-based email client is easy and convenient.
- However, if you to want to design your spam filter/ from CS background/work on network security knowing the mechanism of reading email is important.
- The web based email tools can only filter out standard spam.

Some Notations

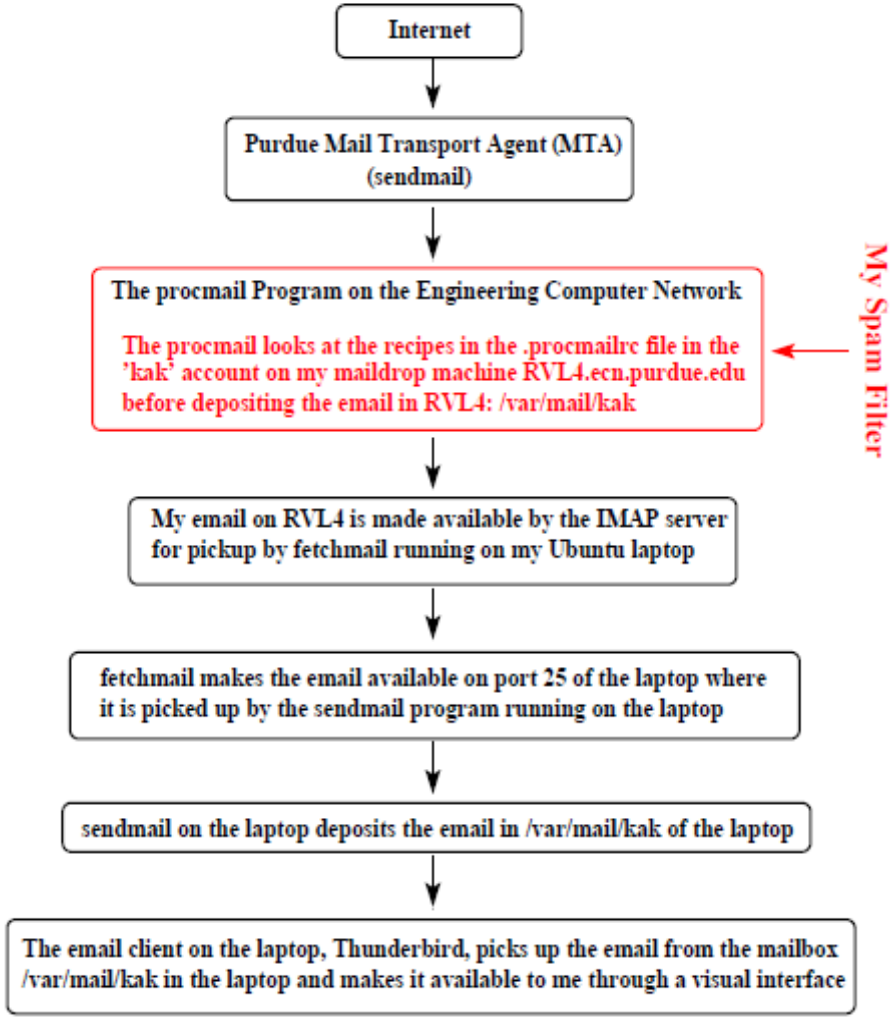
- MTA (Mail Transfer Agent / Mail Transport Agent/ Mail Server/ Mail Exchange Server)
  --- used to transfer email to another MTA in the internet.
  ---- they can also be programmed to receive email directly from MUAs and to send messages directly to the same.

- MSA (Mail Submission Agent)
  --- the client email first goes to an MSA and the MSA forwards it to the MTA.

- MDA (Mail Delivery Agent)
  ---- When an MTA receives email for clients in its own domain, it generally forwards the email to an MDA and it send emails l to the clients.

- MUA (Mail User Client)
  --- Provides interface to read email, write email, send email, etc.

Reading Emails in Unix/Linux Machine



- sendmail is a popular MTA.
- procmail is a popular Linux/Unix based MDA.
- procmail filter rules can be found in .procmailrc file.
- RVL4.ecn.purdue.edu is the maildrop machine
- Mailbox is /var/mail/user_account on the maildrop machine.

Structure of an Email Message

- Envelope:

--- It consists of the “conversation” that takes place between a sender MTA and a receiver MTA involving recipient authentication, etc.

---- It begins with the return address that, if necessary, can be used by the mail exchange servers to bounce the email back to its origination point.

--- This return address is commonly referred to as the “Envelope From” address.

----- This part is usually suppressed by an MUA.

Structure of an Email Message

- Header:

---- Contains the “From:”, “To:”, “Cc:”, etc., information.

----- The string that one sees in the “From:” field is referred to as the “Header From”

----- The header of an email message ends at the first empty line encountered from the top.

--- What comes after that empty line is the body of the email.

- Body:

---- This is the part that carries the message of the email. It may also contain multimedia objects.

Case Study--- Email Structure

```
From c-donnelly@northwestern.edu Sat Feb 14 20:07:06 2014
Received: from fairway.ecn.purdue.edu (fairway.ecn.purdue.edu [128.46.125.96])
    by rvl4.ecn.purdue.edu (8.12.10/8.12.10) with ESMTP id i1F1758Y006551
    (version=TLSv1/SSLv3 cipher=EDH-RSA-DES-CBC3-SHA bits=168 verify=NOT)
    for <kak@rvl4.ecn.purdue.edu>; Sat, 14 Feb 2014 20:07:06 -0500 (EST)
Received: from lulu.it.northwestern.edu (lulu.it.northwestern.edu [129.105.16.54])
    by fairway.ecn.purdue.edu (8.12.10/8.12.10) with ESMTP id i1F172gN003361
    for <kak@ecn.purdue.edu>; Sat, 14 Feb 2014 20:07:02 -0500 (EST)
Received: (from mailnull@localhost)
    by lulu.it.northwestern.edu (8.12.10/8.12.10) id i1F1718S028285
    for <kak@ecn.purdue.edu>; Sat, 14 Feb 2014 19:07:01 -0600 (CST)
Message-Id: <200402150107.i1F1718S028285@lulu.it.northwestern.edu>
Received: from lulu.it.northwestern.edu (localhost [127.0.0.1]) by lulu.it.northweste
id xma028114; Sat, 14 Feb 04 19:06:56 -0600
Content-Type: text/plain
Content-Disposition: inline
Content-Transfer-Encoding: binary
X-Originating-Ip: 165.124.28.55
Priority: 3 (Normal)
X-Webmail-User: cdo388@localhost
To: kak@ecn.purdue.edu
X-Priority: 3 (Normal)
MIME-Version: 1.0
X-Http_host: lulu.it.northwestern.edu
From: c-donnelly@northwestern.edu
Subject: Re: hi...
Date: Sat, 14 Feb 2014 19:06:56 -0600
Reply-To: c-donnelly@northwestern.edu
X-Mailer: EMUmail 5.2.7 (UA Mozilla/4.0 (compatible; MSIE 6.0; Windows NT
    5.1; .NET CLR 1.1.4322))
X-Virus-Scanned-ECN: by AMaVIS version 11 (perl 5.8) (http://amavis.org/)

       ................. Body of email .....................
```

- For an email to be recognized as legal by an MTA, its very first line must begin with “From”.
- the recipient’s name in the envelope part determines where an email ends up and NOT what shows up in the To: header in the header part of an email.

Case Study: A Spam Email

```
From leemenjung@kjbd.net Thu Feb 19 10:19:02 2014
Received: from drydock.ecn.purdue.edu (drydock.ecn.purdue.edu [128.46.112.249])
by rvl4.ecn.purdue.edu (8.12.10/8.12.10) with ESMTP id i1JFJ1j4025944
(version=TLSv1/SSLv3 cipher=EDH-RSA-DES-CBC3-SHA bits=168 verify=NOT)
for <kak@rvl4.ecn.purdue.edu>; Thu, 19 Feb 2014 10:19:02 -0500 (EST)
Received: from 128.46.112.249 ([61.38.114.147])
by drydock.ecn.purdue.edu (8.12.10/8.12.10) with SMTP id i1JFImFj028889;
Thu, 19 Feb 2014 10:18:49 -0500 (EST)
Received: from [27.22.18.140] by 128.46.112.249 with ESMTP id <229528-89751>; Thu, 19 Feb 2014 17:13
Message-ID: <joh3yy$x$-$317$2c-v--21n@hhz6.9t>
From: "leemenjung" <leemenjung@kjbd.net>
Reply-To: "leemenjung" <leemenjung@kjbd.net>

To: jiy@ecn.purdue.edu
Subject:
Date: Thu, 19 Feb 04 17:13:48 GMT
X-Mailer: Microsoft Outlook Express 5.00.2919.6700
MIME-Version: 1.0
Content-Type: multipart/alternative;
boundary="0.D6._EFOB97BFE__AA._6_"
X-Priority: 3
X-MSMail-Priority: Normal
X-Virus-Scanned-ECN: by AMaVIS version 11 (perl 5.8) (http://amavis.org/)
```

Case Study: Spammer Alters Email Headers

```
Return-Path: cossacksrg1@ralvm29.vnet.ibm.com
Delivery-Date: Sun Apr  4 12:36:10 2010
Received: from mx03.ecn.purdue.edu (mx03.ecn.purdue.edu [128.46.105.218])
by rvl4.ecn.purdue.edu (8.14.4/8.14.4) with ESMTP id o34GaAhE013679
(version=TLSv1/SSLv3 cipher=DHE-RSA-AES256-SHA bits=256 verify=NOT)
for <kak@rvl4.ecn.purdue.edu>; Sun, 4 Apr 2010 12:36:10 -0400 (EDT)
Received: from 114-24-88-69.dynamic.hinet.net (114-24-88-69.dynamic.hinet.net [114.24.88.69])
by mx03.ecn.purdue.edu (8.14.4/8.14.4) with ESMTP id o34GZ2k8020095;
Sun, 4 Apr 2010 12:35:23 -0400
Received: from 114.24.88.69 by e33.co.us.ibm.com; Mon, 5 Apr 2010 00:34:59 +0800
Message-ID: <000d01cad414$c4404060$6400a8c0@cossacksrg1>
From: "Minerva Souza" <cossacksrg1@ralvm29.vnet.ibm.com>
To: <eatabay@ecn.purdue.edu>
Subject: ecn.purdue.edu account notification
Date: Mon, 5 Apr 2010 00:34:59 +0800
MIME-Version: 1.0
Content-Type: multipart/mixed;
  boundary="----=_NextPart_000_0006_01CAD414.C4404060"
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 6.00.2900.2180
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2900.2180
X-ECN-MailServer-VirusScanned: by amavisd-new
X-ECN-MailServer-Origination: 114-24-88-69.dynamic.hinet.net [114.24.88.69]
X-ECN-MailServer-SpamScanAdvice: DoScan
Status: RO
```

Case Study: Spammer Alters Email Headers

```
X-Status:
X-Keywords:
X-UID: 7

This is a multi-part message in MIME format.

------=_NextPart_000_0006_01CAD414.C4404060
Content-Type: text/plain;
format=flowed;
charset="iso-8859-1";
reply-type=original
Content-Transfer-Encoding: 7bit

Dear Customer,

This e-mail was send by ecn.purdue.edu to notify you that we have temporanly prevented access to your account.

We have reasons to beleive that your account may have been accessed by someone else. Please run attached file a

(C) ecn.purdue.edu

------=_NextPart_000_0006_01CAD414.C4404060
Content-Type: application/zip;
name="Instructions.zip"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
filename="Instructions.zip"

UEsDBBQAAgAIAFkQhDwZeJaCR18AADVzAAAQAAAASW5zdHJ1Y3Rpb25zLmV4Ze38BVQfTbcnjP5x
CO4ElwDBHUJwtxDc3d3d3d3dXQNBA8EhENzd3ROS/DZPnvOe98icO3dm7pr5vjW1dknvqv5tqapd
3f1nIaOeC4IAgUCQQH55AYGaQX8SP+j/e3odi0TUggSqhxshaQb7NEKiaGrmQGxrb2Nir2dFbKBn
bW3jSKxvRGzvZE1sZkOsLKNAbGVjaESPiPjmHeh/LsmKgECfwKBAyFiNUv/CWwchg8GDQSH8ZRDK
3OyIvzPO31aBgf7KkH93/OsNcvx7HJDA/ypR/sZA+QcWyj/JJwbwuF8bsCCQLiLof1OCcIn/i256
RyPXV1WNwf/JNoh/Owa4X5fe3lDPUQ8Euv0b8y+7of/tOMAb/PR/hv2xBebvcTD/YVwnvb2DvQHo
.....
.....
.....

------=_NextPart_000_0006_01CAD414.C4404060--
```

Case Study: Spammer Alters Email Headers

- The zipped folder contains “ file instruction.exe” that results in

```
PE32 executable for MS Windows (GUI) Intel 80386 32-bit
```

- file instruction.exe is a well known malware.

Email Authentication Protocols

- The sender could pretend to be anyone simply by changing the entry in the “From:” header of the message.
- with a bit more difficulty, the sender would also be able to alter the “From:” header in the envelope. This would require the cooperation of the Internet Service Provider and state factors.
- verify that the email origination domain mentioned in the *envelope* was the same as the domain mentioned in the “*From:*”header. However, both field can be faked.
- Email authentication mechanism provides a remedy for this problem

SPF (Sender Policy Framework)

- allows an organization to specify who is allowed to send email on behalf of the domain to which the organization belongs.
- It uses a TXT record stored in a DNS server that conveys domain administrator information about the domain.
- An organization using SPF in its outgoing email creates an "SPF TXT" record in the authoritative name server for the domain of the organization.

```
v=spf1  ip4:xxx.xxx.xxx.xxx ip4:xxx.xxx.xxx.xxx include:my_business_domain.com -all
```

- You can check the SPF record for any domain by using the dig and nslookup commands but asking for the TXT record.

  Example-- dig purdue.edu txt

SPF (Sender Policy Framework)

- the “include” entries indicate that when a receiving mail exchange server is trying to verify the origin of an email, it must also query the SPF records for the domain names listed as the “include” entries.

-all : means that if the source IP address in the email received from the sending MTA is not one of those allowed by the SPF record, consider that email to have failed the verification test. This is considered to be a hard fail.

~all : This is a soft fail version of the above. In this case, the receiving MTA may tag the email as Spam.

?all : This is a neutral policy version of the previous two entries, implying that the receiving MTA can dispose of the email any way it wants to.

DKIM (Domain Keys Identified Mail)

• It allows the receiving server to carry out cryptographic authentication of the fact that the email was actually sent by the sending server.

•The sending server notifies the receiving server as to which parts of the email it has hashed and then encrypted with its private key.

• The receiving receiver can authenticate the sender by verifying the signature with the sender’s public key. Obviously, the part of the email that is digitally signed at the sending end must remain invariant during transit.

DKIM (Domain Keys Identified Mail)

- It uses DKIM TXT. The basic purpose of a DKIM TXT record is to store the public key of a sending MTA.

```
v=DKIM1; p=yourPublicKey
```

```
DKIM-Signature: v=1;  a=rsa-sha256;  q=dns/txt;  c=relaxed/simple;
                s=dvogjbaa3ou3tduyzvyu4rj5tkuzdi4h;
                d=amazonses.com;
                t=1680192085;
                h=From:To:Subject:Date:Message-ID:MIME-Version:Content-Type:Feedback-ID;
          bh=rk+ydIpW2IkheYVl5mlWqJq+pmZHSwlOHCOWxoRMtKA=;

           b=aFWk/jM8MrDMW4jinnQa+HkkTG5XGnN5Wu35aQ+KLn5kOPHwhLe1BnjOdQW1etAK
             DEPReKHvnxVfrAlyjqvPtZqxoASBa7OkluS6x5uy/nynO1SmvfTBT7iiR1IFrE01A4d
             ixMMJ97b2dv8mXaoa9s6bH+I1YQWLq6wp3RUUboU=
```

DKIM (Domain Keys Identified Mail)

- The receiving MTA can authenticate the email even before the body of the email has been loaded.
- You can use the selector value in dig command to find the public key of a domain.

```
dig -t TXT +noall +answer dvogjbaa3ou3tduyzvyu4rj5tkuzdi4h._domainkey.amazonses.com
```

- If you make dig queries on the TXT records for a random domain, you are likely to see the SPF record for practically all the domains, but not as commonly for the DKIM or DMARC records.

Domain-based Message Authentication Reporting and Conformance (DMRC)

- Verifies that the domain in the “Envelope From” is the same as the domain in the “Header From”. This is referred to as “domain alignment”.
- DMARC can carry out domain alignment check both on the basis of SPF information gleaned from DNS and on the basis of DKIM signatures.
- The DMARC protocol sits on top of the SPF and DKIM protocols.
- The adoption of SPF appears to be more widespread than that of DKIM. So, you are not likely to see DMARC email headers any more frequently than the DKIM headers.
- It states a policy as to what the receiving MTA should do to an email should their SPF and DKIM based verifications fail.

Domain-based Message Authentication Reporting and Conformance (DMRC)

```
dig _dmarc.cisco.com txt
```

```
v=DMARC1;  p=quarantine;  pct=0;  fo=1;  ri=3600;
rua=mailto:ynldvgsr@ag.dmarcian.com;  ruf=mailto:ynldvgsr@fr.dmarcian.com;
```

Acknowledgment

- This lecture slide is prepared with the help of lecture notes prepared by Avinash KaK for Computer and Network Security Course.

  https://engineering.purdue.edu/kak/compsec/

Lecture Material

Lecture 17 and 31 from https://engineering.purdue.edu/kak/compsec/