Today

- network attacks

network attacks

- denial of service (DoS)
- account capture
- impersonation/spoofing
- scan and exploit
- advanced

denial of service

- raw sockets – spoofed source address
- smurf, teardrop, land, SYN floods
- crash or overflow target
- difficult to defend or locate source

account capture

what secret do you have?

- social engineering
- shoulder surfing
- guess
- crack /etc/passwd
- network sniffer – reusable passwords

Cuckoo's Egg

1986 LBL breakin

- Detection: account discrepancy
- session traces and honeypot
- How: guest account/emacs bug
- What: root access
  - password guessing and cracking
  - dialout exploit
  - rhost exploit
  - log cleanup
- Why: seeking military secrets
- Who: German hackers/KGB

pursuit and prosecution

remote attacks

probably from compromised host

- map network
  - icmp echo broadcast
  - DNS zone transfer
  - sequential scan
- port scans (nmap)
- OS id scan
- port probe
- service exploit
  - bug (sendmail, web server)
  - buffer overflow

buffer overlow

- poor programming
- strcpy, scanf, sprintf, strcat, gets
- hacker scans sources for possible bad coding
- overflow stack, give you a remote root shell
- popd, sendmail, amd, bind, imapd, samba, CDE, statd, IIS, ...
- recent attacks on RPC services, bypass port 111 filters and port scan for RPC programs NULL proc.

once you're in

- OS exploit for root
- rootkit
- backdoor
- sniffer
- get /etc/passwd
- exploit rhosts
- return to get sniffer logs
- tell your friends

Morris Worm

Attacked ORNL November, 1988

- widespread Internet attack
- 6000 hosts (10%)
- Detection: system console log
- How: sendmail or buffer overflow
- What: root access, self-spawning
- contained at ORNL, dumb luck
- Why: experimenting
- Who: Cornell student

prosecuted

host impersonation

How do I spoof thee?
Let me count the ways

- boot with Bob's IP
- ARP poisoning (hunt)
- exploit trusted host (rhosts)
  - your own DNS
  - DNS poisoning
  - source routing
- spoofed source address and sequence number guessing

BSD net security

- based on trusted hosts
- hosts.equiv and .rhosts (r utilities)
- user and host impersonation threat
- notion of reserved port for trusted servers

reserved ports

- must be super-user to listen() on ports < 1023
- prevent nonprivileged user from impersonating well-known service (rlogind, ftpd, telnetd)
- lpr/lpd use (**rresvport()**) and confirm reserved port
- just a convention, no RFC requirement
- PC or superuser can easily impersonate

ruserok.c

- r-utilities check */etc/hosts.equiv* or *.rhosts*
- assumes trusted host
- prevents clear-text password from passing over net
- but once you break in to one system, you may get access to many more
- based on user-name and host name
- r-daemons use **ruserok()** to authorize

ruserok.c

```
ruserok(rhost, superuser, cliuname, servuname)
char  *rhost;        /* "hostname" or "hostname.domain" of client
int   superuser;    /* 1 if caller wants to be root on this (server'
char  *cliuname;    /* username on client's system */
char  *servuname;   /* username on this (server's) system */
{
   hostfp = superuser ? (FILE *) 0 : fopen("/etc/hosts.equiv", "r");
   first = 1;
again:
   if (hostfp) {
      if (_validuser(hostfp, clihostname, servuname, cliuname,
                     hostlen) == 0) {
         fclose(hostfp);      /* all OK, close file */
         return(0);     /* we're done */
      }
      fclose(hostfp);      /* not valid, close file */
   }
   if (first == 1 && (_check_rhosts_file || superuser)) {
      struct stat statbuff;
      struct passwd  *pwd;
      char     buff[MAXPATHLEN];
      first = 0;
      if ( (pwd = getpwnam(servuname)) == NULL)
         return(-1); /* no password file entry */
      strcpy(buff, pwd->pw_dir); /* get home directory of servuname
      strcat(buff, "/.rhosts");  /* will be "//.rhosts" if root */
      if ( (hostfp = fopen(buff, "r")) == NULL)
         return(-1); /* can't open user's .rhosts file */
```



```
   if ((fstat(fileno(hostfp), &statbuff) < 0) ||
       (statbuff.st_uid != 0 && statbuff.st_uid != pwd->pw_uid) ||
       (statbuff.st_mode & 022)) {
      fclose(hostfp);
      return(-1);
   }
   goto again; /* go and call _validuser() */
}
return(-1);    /* not a valid user */
```



```
_validuser(hostfp, clihostname, servuname, cliuname, hostlen)
FILE  *hostfp; /* FILE pointer to "/etc/hosts.equiv", ".rhosts",
              or "/etc/hosts.lpd" file */
char  *clihostname;  /* client's "hostname" or "hostname.domain" */
char  *servuname; /* username on this (server's) system */
char  *cliuname;  /* username on client's system */
int   hostlen; /* -1 if "hostname"; else its "hostname.domain" and t
              is the #chars in hostname */
{
   while (fgets(hostname, sizeof(hostname), hostfp)) {
      ptr = hostname;
      while (*ptr != '\n' && *ptr != ' ' &&
             *ptr != '\t' && *ptr != '\0') {
            *ptr = isupper(*ptr) ? tolower(*ptr) : *ptr;
            ptr++;
      }
      if (*ptr == ' ' || *ptr == '\t') {
...      user = ptr; /* user name in file */
     ....
      if (_checkhost(clihostname, hostname, hostlen) &&
          (strcmp(cliuname, *user ? user : servuname) == 0) )
         return(0);  /* OK */
   }
   return(-1);    /* end-of-file, no match found */
}
```

```
_checkhost(clihostname, hostfield, hostlen)
char  *clihostname;  /* client's "hostname" or "hostname.domain" */
char  *hostfield; /* the hostname field from the file */
int   hostlen; /* -1 if no ".domain", else length of "hostname" */
{
   if (hostlen == -1)
      return(strcmp(clihostname, hostfield) == 0); /* 1 if equal */
...  /* check various domain name combo's */
   return(strcmp(locdomptr, clihostname + hostlen + 1) == 0);
               /* returns 1 if equal, 0 otherwise */
}
```


trust fixes

network/host

- router block external packets with internal host address
- block source routing
- replace inetd with xinetd
- tcpwrappers (tcpd)
  - free, no changes to apps or inetd
  - inetd services only
  - allow/deny per service
  - double DNS
  - deny source routing
  - log and alarm

wrappers

```
/etc/inetd.conf
 ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  wu.ftpd
 telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
 shell   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rshd -L
 login   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rlogind

/etc/hosts.deny
 in.rlogind: ALL
 in.telnetd: ALL
 in.rshd: ALL
 wu.ftpd: ALL

/etc/hosts.allow
 in.rlogind: 128.219., 134.167., 127.
 in.telnetd: 128.219., 134.167., 127.
 wu.ftpd: 128.219., 134.167., 127.
 in.rshd: 128.219., 134.167.
```


Mitnick attack

sophisticated attack at SDSC, 1994

- Detection: system logs
- How: IP spoofing, sequence number guessing, phone switches, rhosts
- What: root access
- Why: steal files (cell phone software)
- Who: Kevin Mitnick

prosecuted

sequence number guessing

- fixed increment of "new" sequence numbers
- probe target to deduce next sequence number
- take out trusted host
- spoof trusted host to target host with raw socket packets
- you know flow of session
- you don't get server packets

router blocks local from external

don't base trust on IP address or name

session hijacking

advanced

- bad guy in path of hosts
- sniff initial session establishment
- reset client and take over session
- can hijack strong-authenticated session (skey,securid)

avoid intrusion detection with IP frags, bad TCP checksums, TCP segment overlap, ...

next time

IP defenses