Remote incidents

- Packets traversing network stack
- Remote or local information?
- Very difficult to erase remote network information, very little information kept
- Gathering local network information
- Important nodes on network
- Initial vs. subsequent connections
What we won’t Cover

• DOS - smurf/ping-O-death/winnuke/etc.
• Detection of sniffers
Typical Network Attack
Eradicating Network Traces

- Virtually impossible in most cases
- Don’t know where data was saved
- Must determine where data flow went
- Compromise all routers, hosts, etc.
- Destroy all information there, plus recursively follow this list
Gathering Information

- System configuration
- System & user programs
- System & kernel memory
- Raw mem/disk - anything with IP
  #’s/hostnames
System Configuration (1/2)

- Enter into the realm of auditing
- Invisible changes
- Freezing system should *gather* most of this
- Need to know how system *should* look like
- Kernel
- Packet filters
System Configuration (2/2)

- Access control (hosts.allow, httpd.conf, sshd_config, etc.)
- Trust (servers, rhosts, network info, etc.)
- Configs (routes, inetd.conf, startup files, etc.)
- Protocols
- Userland (.rhosts, .forward, etc.)
Programs

- Queries to system
- Program memory
- Logs
Queries to the System

- netstat(8c)
- arp(8c)
- lsof
- portscanners
### Netstat - Show Net Status

```
% netstat -a -f inet

Active Internet connections (including servers)
Proto R-Q S-Q Local Address       Foreign Address         state
tcp  0  0  flying.smtp           192.215.43.108.4778   EST
tcp  0  0  flying.http           dialup6929.nssl..2787  EST
tcp  0  0  flying.smtp           192.215.43.108.4769   WAIT
tcp  0  0  flying.http           telapex..2198          SYN_RCVD
```
# Netstat, etc.

```bash
% netstat -rn
```

Routing tables

<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway</th>
<th>Flags</th>
<th>Refcnt</th>
<th>Use</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>UH</td>
<td>1</td>
<td>1365</td>
<td>lo0</td>
</tr>
<tr>
<td>default</td>
<td>209.179.181.129</td>
<td>UG</td>
<td>17</td>
<td>2089112</td>
<td>le0</td>
</tr>
</tbody>
</table>
**arp** - Address Resolution
Display and Control

- What ethernet is claiming IP address
- Only useful on LAN
- Easy to forge
- Can give system types

Portscanners

• % tcp_scan fish.com 1-1024
21: ftp
23: telnet
25: smtp
53: domain
515: printer
667: UNKNOWN
Logs

- Syslog
- NFS
- NIS
- DNS
- Kernel
Every Scrap of Data - `bind`

- Keeps track of **EVERY** query of host
- Send a **SIGINT** signal to `bind`
- Dumps database into `named_dump.db`
- Compare vs. system logs, known hosts, use TTL vs. time left in memory
- A few megabytes of fun…
- $10^4 << 10^8!$
More Scraps…

- Use `pcat` to dump a processes memory
- If program (esp. auth daemons) talks to net, possibly has net info
- Even if doesn’t log, it remembers!
- Good for system, great vs. intruder tools
- Easy to spot hosts
- Only reasonable way to prevent is to kill daemon, restart (might see PID change)
Using pcat to Examine Memory

- **ps/lsol** locates program (*nfsd, statd, etc.*)
  
  ```
  # pcat 123 | strings > 123.mem
  ```

- **grep ‘[host/IP pattern]’ 123.mem**

- **strings & less** to further examine

(Also **/dev/mem & /dev/kmem**)
Gathering Remote Information

- Speed is important!
- Hosts - recursively freeze each
- Routers & Access equipment
- Telcos
- ISPs
- FIRST/CERTs, etc.
Routers, etc.

- As complicated as hosts… and less documented & understood
- Can seriously impact investigation
- Lots of ways to manage & examine
- All do things differently
- Should look at:
  - Routes
  - Arp/IP/etc tables
  - Any network information