Intruders

- significant issue for networked systems is hostile or unwanted access
- either via network or local
- can identify classes of intruders:
  - masquerader
  - misfeasor
  - clandestine user
- varying levels of competence
  - key goal often is to acquire passwords
Password Guessing

- one of the most common attacks
- attacker knows a login (from email/web page etc)
- then attempts to guess password for it
  - try default passwords shipped with systems
  - try all short passwords
  - then try by searching dictionaries of common words
  - intelligent searches try passwords associated with the user (variations on names, birthday, phone, common words/interests)
  - before exhaustively searching all possible passwords
- success depends on password chosen by user
- surveys show many users choose poorly
Password Capture

• another attack involves password capture
  – watching over shoulder as password is entered
  – using a Trojan horse program to collect
  – monitoring an insecure network login (eg. telnet, FTP, web, email)
  – extracting recorded info after successful login (web history/cache, last number dialed etc)
Intrusion Detection

Figure 18.1  Profiles of Behavior of Intruders and Authorized Users
Approaches to Intrusion Detection

• statistical anomaly detection
  – threshold
  – profile based

• rule-based detection
  – Anomaly, based on previous usage pattern
  – penetration identification
Audit Records

• fundamental tool for intrusion detection
• native audit records
  – part of all common multi-user O/S
• detection-specific audit records
  – created specifically to collect wanted info
Statistical Anomaly Detection

• threshold detection
  – count occurrences of specific event over time
  – if exceed reasonable value assume intrusion
  – alone is a crude & ineffective detector

• profile based
  – characterize past behavior of users
  – detect significant deviations from this
  – profile usually multi-parameter
Audit Record Analysis

• foundation of statistical approaches
• analyze records to get metrics over time
  – counter, gauge, interval timer, resource use
• use various tests on these to determine if current behavior is acceptable
  – mean & standard deviation, multivariate, markov process, time series, operational
### Table 18.1 Measures That May Be Used for Intrusion Detection

<table>
<thead>
<tr>
<th>Measure</th>
<th>Model</th>
<th>Type of Intrusion Detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login frequency by day and time</td>
<td>Mean and standard deviation</td>
<td>Intruders may be likely to log in during off-hours.</td>
</tr>
<tr>
<td>Frequency of login at different locations</td>
<td>Mean and standard deviation</td>
<td>Intruders may log in from a location that a particular user rarely or never uses.</td>
</tr>
<tr>
<td>Time since last login</td>
<td>Operational</td>
<td>Break-in on a &quot;dead&quot; account.</td>
</tr>
<tr>
<td>Elapsed time per session</td>
<td>Mean and standard deviation</td>
<td>Significant deviations might indicate masquerader.</td>
</tr>
<tr>
<td>Quantity of output to location</td>
<td>Mean and standard deviation</td>
<td>Excessive amounts of data transmitted to remote locations could signify leakage of sensitive data.</td>
</tr>
<tr>
<td>Session resource utilization</td>
<td>Mean and standard deviation</td>
<td>Unusual processor or I/O levels could signal an intruder.</td>
</tr>
<tr>
<td>Password failures at login</td>
<td>Operational</td>
<td>Attempted break-in by password guessing.</td>
</tr>
<tr>
<td>Failures to login from specified terminals</td>
<td>Operational</td>
<td>Attempted break-in.</td>
</tr>
</tbody>
</table>

#### Command or Program Execution Activity

<table>
<thead>
<tr>
<th>Measure</th>
<th>Model</th>
<th>Type of Intrusion Detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution frequency</td>
<td>Mean and standard deviation</td>
<td>May detect intruders, who are likely to use different commands, or a successful penetration by a legitimate user, who has gained access to privileged commands.</td>
</tr>
<tr>
<td>Program resource utilization</td>
<td>Mean and standard deviation</td>
<td>An abnormal value might suggest injection of a virus or Trojan horse, which performs side-effects that increase I/O or processor utilization.</td>
</tr>
<tr>
<td>Execution denials</td>
<td>Operational</td>
<td>May detect penetration attempt by individual user who seeks higher privileges.</td>
</tr>
</tbody>
</table>

#### File access activity

<table>
<thead>
<tr>
<th>Measure</th>
<th>Model</th>
<th>Type of Intrusion Detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read, write, create, delete frequency</td>
<td>Mean and standard deviation</td>
<td>Abnormalities for read and write access for individual users may signify masquerading or browsing.</td>
</tr>
<tr>
<td>Records read, written</td>
<td>Mean and standard deviation</td>
<td>Abnormality could signify an attempt to obtain sensitive data by inference and aggregation.</td>
</tr>
<tr>
<td>Failure count for read, write, create, delete</td>
<td>Operational</td>
<td>May detect users who persistently attempt to access unauthorized files.</td>
</tr>
</tbody>
</table>
Base-Rate Fallacy

• practically an intrusion detection system needs to detect a substantial percentage of intrusions with few false alarms
  – if too few intrusions detected -> false security
  – if too many false alarms -> ignore / waste time
• this is very hard to do
• existing systems seem not to have a good record
Distributed Intrusion Detection Architecture
Honeypots

- decoy systems to lure attackers
  - away from accessing critical systems
  - to collect information of their activities
  - to encourage attacker to stay on system so administrator can respond

- are filled with fabricated information
Password Management

- front-line defense against intruders
- users supply both:
  - login – determines privileges of that user
  - password – to identify them
- passwords often stored encrypted
  - Unix uses multiple DES (variant with salt)
  - more recent systems use hash function
Managing Passwords

- need policies and good user education
- protect password file from general access
- Enforce rules for “good” passwords
- Change password periodically
- Run password –guessing program
- Monitor login failures
- Proactive Password Checking
Chapter 19 – Malicious Software

*What is the concept of defense: The parrying of a blow. What is its characteristic feature: Awaiting the blow.*

—*On War, Carl Von Clausewitz*
Viruses and Other Malicious Content

• computer viruses have got a lot of publicity
• one of a family of malicious software
• effects usually obvious
• have figured in news reports, fiction, movies (often exaggerated)
• getting more attention than deserve
• are a concern though
Logic Bomb

• one of oldest types of malicious software
• code embedded in legitimate program
• activated when specified conditions met
  – eg presence/absence of some file
  – particular date/time
  – particular user
• when triggered typically damage system
  – modify/delete files/disks
Trojan Horse

• program with hidden side-effects
• which is usually superficially attractive
  – eg game, s/w upgrade etc
• when run performs some additional tasks
  – allows attacker to indirectly gain access they do not have directly
• often used to propagate a virus/worm or install a backdoor
• or simply to destroy data
Zombie

- program which secretly takes over another networked computer
- then uses it to indirectly launch attacks
- often used to launch distributed denial of service (DDoS) attacks
- exploits known flaws in network systems
Viruses

• a piece of self-replicating code attached to some other code
  – cf biological virus

• both propagates itself & carries a payload
  – carries code to make copies of itself
  – as well as code to perform some covert task
Virus Operation

• virus phases:
  – dormant – waiting on trigger event
  – propagation – replicating to programs/disks
  – triggering – by event to execute payload
  – execution – of payload
Virus Structure

program V :=
  {goto main;
   1234567;
   subroutine infect-executable := {loop:
   file := get-random-executable-file;
   if (first-line-of-file = 1234567) then goto loop
   else prepend V to file; }
   subroutine do-damage := {whatever damage is to be done}
   subroutine trigger-pulled := {return true if some condition holds}
   main: main-program := {infect-executable;
   if trigger-pulled then do-damage;
   goto next;}

   next:
Macro Virus

- **macro code** attached to some **data file**
- interpreted by program using file
  - eg Word/Excel macros
  - esp. using auto command & command macros
- code is now platform independent
- is a major source of new viral infections
Email Virus

• spread using email with attachment containing a macro virus
• triggered when user opens attachment
• or worse even when mail viewed by using scripting features in mail agent
• usually targeted at Microsoft Outlook mail agent & Word/Excel documents
Worms

• replicating but not infecting program
• typically spreads over a network
Worm Operation

• worm phases like those of viruses:
  – dormant
  – propagation
    • search for other systems to infect
    • establish connection to target remote system
    • replicate self onto remote system
  – triggering
  – execution
Virus Countermeasures

• viral attacks exploit lack of integrity control on systems
• to defend need to add such controls
• typically by one or more of:
  – **prevention** - block virus infection mechanism
  – **detection** - of viruses in infected system
  – **reaction** - restoring system to clean state
Anti-Virus Software

- **first-generation**
  - scanner uses virus signature to identify virus
  - or change in length of programs
- **second-generation**
  - uses heuristic rules to spot viral infection
  - or uses program checksums to spot changes
- **third-generation**
  - memory-resident programs identify virus by actions
- **fourth-generation**
  - packages with a variety of antivirus techniques
  - eg scanning & activity traps, access-controls
Advanced Anti-Virus Techniques

• generic decryption
  – use CPU simulator to check program signature & behavior before actually running it

• digital immune system (IBM)
  – general purpose emulation & virus detection
  – any virus entering org is captured, analyzed, detection/shielding created for it, removed
Firewalls – Packet Filters

(a) Packet-filtering router
## Firewalls – Packet Filters

### Table 20.1 Packet-Filtering Examples

<table>
<thead>
<tr>
<th>action</th>
<th>ourhost</th>
<th>port</th>
<th>theirhost</th>
<th>port</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>block</td>
<td>*</td>
<td>*</td>
<td>SPIGOT</td>
<td>we don’t trust these people</td>
</tr>
<tr>
<td></td>
<td>allow</td>
<td>OUR-GW</td>
<td>25</td>
<td>*</td>
<td>connection to our SMTP port</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>action</th>
<th>ourhost</th>
<th>port</th>
<th>theirhost</th>
<th>port</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>block</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>default</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>action</th>
<th>ourhost</th>
<th>port</th>
<th>theirhost</th>
<th>port</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>allow</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>action</th>
<th>src</th>
<th>port</th>
<th>dest</th>
<th>port</th>
<th>flags</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>allow</td>
<td>{our hosts}</td>
<td>*</td>
<td>*</td>
<td>25</td>
<td>our packets to their SMTP port</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>their replies</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>action</th>
<th>src</th>
<th>port</th>
<th>dest</th>
<th>port</th>
<th>flags</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>allow</td>
<td>{our hosts}</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>our outgoing calls</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>replies to our calls</td>
</tr>
<tr>
<td></td>
<td>allow</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>ACK</td>
<td>traffic to nonservers</td>
</tr>
</tbody>
</table>
Attacks on Packet Filters

- **IP address spoofing**
  - fake source address to be trusted
  - add filters on router to block

- **source routing attacks**
  - attacker sets a route other than default
  - block source routed packets

- **tiny fragment attacks**
  - split header info over several tiny packets
Firewalls - Application Level Gateway (or Proxy)
Firewalls - Application Level Gateway (or Proxy)

- use an application specific gateway / proxy
- has full access to protocol
  - user requests service from proxy
  - proxy validates request as legal
  - then actions request and returns result to user
- need separate proxies for each service
  - some services naturally support proxying
  - others are more problematic
  - custom services generally not supported
Firewalls - Circuit Level Gateway