# How to do a Security Audit

- Determine type of audit
- Pre-audit: verify your tools and environment
- Audit/review security policy
- Gather audit information
- Generate an audit report
- Take actions based on the report’s findings
- Safeguard data & report

# Determine Type of Audit

- Host
- Firewall
- Network

# Verify your tools and environment

- **The golden rule of auditing**
- Bootstrapping problem
- Audit tools
- The Audit platform

# The Golden Rule of Auditing

Verify **ALL** tools used for the audit are untampered with; if the results of the auditing tools cannot be trusted, the audit is useless.

# The Bootstrapping Problem

If the only way to verify that your auditing tools are ok is by using auditing tools, then...

# Audit Tools - Trust?

- Write them yourself
- Find a trusted source (person, place)
- Verify them with a digital signature (MD5)

# Audit Tools - the Hall of Fame

- COPS/Tiger
- Crack
- SATAN/ISS
- lsof /pff
- tcpdump
- ipsend
- MD5/DES/PGP
- (Insert your own here)

# The Audit Platform

- Should have extraordinary security
- Submit it to a firewall+ type of audit
- Physical access should be required to use
- No network services running

# Choosing a security audit platform

**Hardware**

- laptop computer
- three kilograms or less
- graphics display
- MB memory
- MB disk
- ethernet (as many connectors as possible)
- Thinkpad/Toshiba

# Choosing a security audit platform (cont.)

**Software**

- Unix
- Secured OS
- OS source code
- Audit tools
- Development tools

# Unix

- Linux or NT
- BSD/OS, FreeBSD, and SunOS*
- Source code
- A good development platform
- An older, stable platform
- Large body of available literature

# Audit/review security policy

- Utilize existing or use “standard” policy
- Treat the policy as a potential threat
- Does it have all the basic components?
- Are the security configs comprehensive?
- Examine dissemination procedures

# Treat the policy as a potential threat

- Bad policies are worse than none at all
- Good policies are very rare
- Look for clarity & completeness
- Poor grammar and spelling are not tolerated

# Does it Have All the Basic Components?

- Who can use resources
- Proper use of the resources
- Granting access & use
- System Administrator privileges
- User rights & responsibilities
- What to do with sensitive information

# Are the security configs comprehensive?

- Details are important!
- Addresses specific technical problems (COPS-like tests, network services run, etc.)
- Allowable trust must be clearly outlined
- Should specify specific tools (The TCP wrappers, S/Key, etc.) that are used
- Must have explicit time schedules of security audits and/or tools used
- **Logfiles** must be regularly examined!

# Examine dissemination procedures

- Policies are worthless unless people read and understand them
- Ideally it is distributed and addressed when people join org.
- E-mail is useful for updates, changes
- **Written** user acknowledgment necessary

# Gather audit information

- Talk to/Interview people
- Review Documentation
- Technical Investigation

# Talk to/Interview people

- Difficult to describe, easy to do
- Usually ignored
- Users, operators, SA’s, janitors, managers...

# Talk to/Interview people (cont.)

- Usage & patterns
- Have they seen/read the security policy?
- What can/can't they do, in own words
- Could they get root/system privs?
- What are systems used for?
- What are the critical systems?
- How do they view the security audit?

# Review Documentation

- Hardware/software inventory
- Network topology
- Key personnel
- Emergency numbers
- Incident logs

# Technical Investigation

- Run static tools (COPS, Crack, etc.)
- Check system logs
- Check system against known vuls (CERT, bugtraq, CIAC advisories, etc.)
- Follow startup execution
- Check static items (config files, etc.)
- Search for privileged programs (SUID, SGID, run as root)
- Examine all trust

# Technical Investigation (cont.)

- Check extra network services (NFS, news, httpd, etc.)
- Check for replacement programs (wu-ftpd, TCP wrappers, etc.)
- Code review “home grown” programs (CGI’s, finger FIFO’s, etc.)
- Run dynamic tools (ps, netstat, lsof, etc.)
- Actively test defenses (packet filters, TCP wrappers, etc.)

# Run Static Tools

- COPS/Tiger
- SATAN/ISS
- Crack
- Fuck

# Follow Startup Execution

- Boot (P)ROMS
- init
- Startup programs (rc.* like files)

# Check static items

- Examine all config files of running processes (inetd.conf, sendmail.cf, etc.)
- Examine config files of programs that can start up dynamically (ftpd, etc.)

# Search for privileged programs

- Find all SUID/SGID programs
- Look at all programs executed as root
- Examine:
  - Environment
  - Paths to execution
  - Configuration files

# Examine all Trust

- rhosts, hosts.equiv
- NFS, NIS
- DNS
- Windowing systems
- User traffic and interactive flow

# Check Extra Network Services

- NFS/AFS/RFS
- NIS
- News
- WWW/httpd
- Proxy (telnet, ftp, etc.)
- Authentication (Kerberos, security tokens, special services)
- Management Protocols (SNMP, etc.)

# Check for replacement programs

- wu-ftpd
- TCP wrappers
- Logdaemon
- Xinetd
- GNU fingerd

# Code review “home grown”/non-standard programs

- Network daemons
- Anything SUID, SGID
- Programs run as system account
- CGI’s
- finger FIFO’s

# Code review, etc. (cont.)

- Bad signs:
  - external commands (system, shell, etc.)
  - /usr/ucb/mail
  - large size
  - No documentation
  - No comments in code
  - No source code available

# Run dynamic tools

- ps
- netstat
- top
- lsof/pff

# Actively test defenses

- packet screens
- TCP wrappers
- Other defense programs

# Safeguard Data & Report

- Save for the next audit
- Do **<u>not</u>** keep on-line
- Use strong encryption if stored electronically
- Limit distribution to those who “need to know”
- Print out report, sign, and number copies
- Keep physically secure from intruders & natural disasters