SUMMARY PAPER

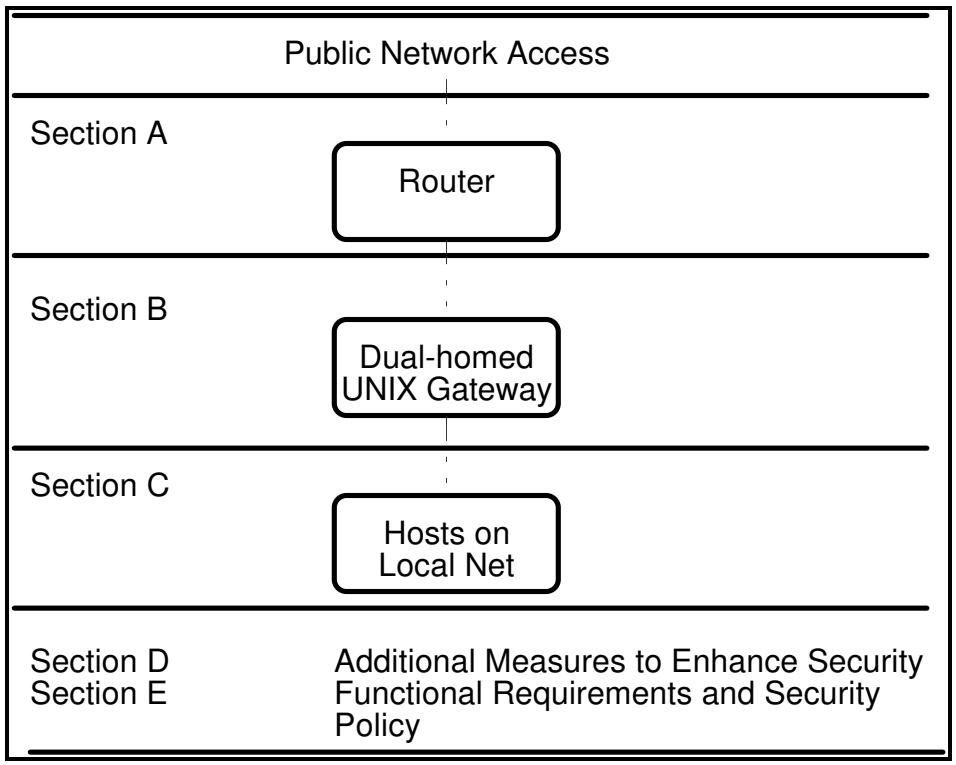
An Architectural Overview of UNIX Network Security

(Specifically oriented toward Internet connectivity)

By Robert B. Reinhardt, September 19, 1992
(breinhar%srg@uunet.uu.net) - work
(breinhar@access.digex.com) - home

Nothing in this paper should be construed as a product endorsement. The contents herein are the a result of light research and some prototype implementation that I've done over the past nine months. I don't know if this will help you or not. This is basically just a digest of information that a lot of people already know. This is for those of you who don't already know.

For each of the FIREWALL layers (sections) below, there is a subsection that follows that gives a brief description of some of the most widely used tools and techniques for implementing security controls and their availability.



Before starting into a description of the various elements of each of the above layers I feel I should reiterate the need for first developing a local security policy. Each organization or site needs to have an effective security policy. There are many tools and techniques available to implement security controls, but you should first conduct a thorough analysis of what your needs are, in order to design and implement an efficient operational environment. You need to determine what your requirements for network services and features are, what level of security

you require, and what risks you are willing to accept. Sometimes the benefit outweighs the risk, sometimes not. But, those decisions differ for each organization. The "firewall" concept for creating a security demarkation point between your local net and the outside, as well as the various methods for enhancing security may not be appropriate for everyone, and in some cases may not go far enough. But I believe this a good starting point for almost anyone with a general concern for UNIX network security.

Let me apologize right now to the authors of these tools and designs. Since I am just giving a brief overview, I cannot do justice to a complete description of them. To the reader let me say that you should check the availability section and whenever possible obtain a README or other information before making your decisions. In many cases there is at least one paper (in most cases probably several) that have been published that describe these things in better detail. I'll try to list at least one source for each.

Papers describing most if not all of these packages and techniques can be found in the SYMPOSIUM PROCEEDINGS of the USENIX Third UNIX Security Symposium (c) 1992 by the USENIX Association.

Some of the functionality of these tools overlap. Since you have the source to these tools, you can modify them or customize them to add new features.

SECTION A - Physical Access to your Network

A1. Packet filtering. Several internet protocol routers provide the capability to filter packets. Packet filtering allows you to program the router to make a decision whether or not traffic can pass to or from your network based on several criteria such as: source ip address, destination ip address, protocol, tcp or udp port, etc.

Availability: I only have experience with CISCO routers, however I've been told that Wellfleet and Proteon routers also have this feature. There may be other vendors as well, but they probably all implement it a little differently. Read: Smoot Carl-Mitchell and John S. Quarterman, "Building Internet Firewalls"; UnixWorld; February, 1992; pp 93-102.

NOTE: This layer of your security firewall also includes other methods of access between networks such as CALL-BACK MODEMS.

SECTION B - Logical Access to your Network

B1. TCP_WRAPPER. The "TCP_WRAPPER" tool provides monitoring and control of network services. Essentially, what happens is that you configure inetd on your dual-homed gateway to run the TCP WRAPPER software whenever certain services (ports) are connected to. Depending on how you configure TCP WRAPPER, it will then LOG information about the connection and then actually start the intended SERVER program that the connection was intended for. Since you have the source to the tool, you can modify it to do more depending on what your needs are. For example, you may want TCP WRAPPER to connect the user to a proxy service instead of the actual program, then have your proxy software handle the transaction in whatever way your security requirements demand.

Availability: This is available from several sources, but to ensure that you get the most recent copy that CERT has verified, you should use anonymous FTP to retrieve it from cert.org in ~/pub/tools/tcp_wrappers/tcp_wrappers.*.

B2. SOCKS Library and sockd. The "sockd" and "SOCKS Library" provide another way to implement a "tcp wrapper." It is not intended to make the system it runs on secure, but rather to centralize ("firewall") all external internet services. The sockd process is started by inetd

whenever a connection is requested for certain services, and then only allows connections from approved hosts (listed in a configuration file). The sockd also will LOG information about the connection. You can use the Socks Library to modify the client software to directly utilize the sockd for outgoing connections also, but this is described as very tedious and of course requires you to have the source to those client programs.

Availability: Unknown. Contact the authors for more information. David Koblas (koblas@sgi.com) or Michelle R. Koblas (mkoblas@nas.nasa.gov).

B3. Kernel_Wrap for SunOS/RPC via Shared Libraries. Essentially this is a wrapper for SunOS daemons that use RCP, such as portmap, ypserv, ypbind, ypupdated, mountd, pwdauthd, etc. To utilize this, you must have SunOS 4.1 or higher and must have the capability to rebuild your shared libraries (but, you don't need the source to your entire system). Essentially what happens is that you modify the function calls that the kernel uses to establish RPC connections, such as accept(), recvfrom() and recvmsg(). Since these calls are maintained in the shared libraries, you have access to modify them without rewriting the kernel.

Availability: The secured C library package to implement this is available via anonymous FTP from eecs.nwu.edu in ~/pub/securelib.

B4. SWATCH. Simple WATCHER is really two things, it is a program used to parse through the myriad of LOG data generated by the various security programs, in particular "syslog." But, it's more than that. It is fully configurable with triggers (actions), so that while it is continuously monitoring the LOG in "real-time," it can take actions based upon certain high-priority events that you tell it to watch for. To get full use of this, you will need to modify your network service daemons such as ftpd and telnetd so that enhanced logging is added to syslog, to feed SWATCH.

Availability: The SWATCH source and documentation is available via anonymous FTP from sierra.stanford.edu in ~/pub/sources.

B5. Controlled Access Point (CAP). This is more of a method or protocol definition than a specific product. CAP provides a network mechanism intended to reduce the risk of: password guessing, probing for well-known accounts with default passwords, trusted host rlogin, and password capture by network snooping. It is really a design for a variation or enhancement to the general firewall approach to connecting two or more networks. In the paper describing this there is an example of two local nets, one a secure segment with an authentication service, and the other an unsecure segment. Both communicate with each other via a CAP, while there is a router for communication to public networks connected on the unsecure side of the CAP. The CAP is essentially a router with additional functionality to detect incoming connection requests, intercept the user authentication process, and invoke the authentication server.

Availability: Unknown. Contact the authors for more information. J. David Thompson (thompsond@orvb.saic.com) and Kate Arndt (karndt@mitre.org).

B6. Mail Gateway. This is more of a procedure than a software package (although there are packages designed just to do this). I included this to maintain continuity with what I'm trying to illustrate in this paper. This really should be applied to all network services that require external connectivity (meaning any communication over non-private or non-secure channels). In the simplest implementation of this, you configure your router to filter packets so that all mail traffic (SMTP protocol for example) is only allowed to and from one host, the "Mail Gateway." Likewise, your DNS and MTA software will need to be configured for this as well.

B7. TTY_WRAPPER. This is one of my pet ideas. I have not seen something like this around, and I'll probably never have time to develop it. But, essentially this would be like "TCP

WRAPPER," only it is designed specifically for serial communications. After that, we will need a "PSEUDO-TTY WRAPPER," but that's for another day.

SECTION C - Physical and Logical Access to Hosts on your Network

C1. Computer Oracle and Password System (COPS). COPS is a UNIX security status checker. Essentially what it does is check various files and software configurations to see if they have been compromised (edited to plant a trojan horse or back door), and checks to see that files have the appropriate modes and permissions set to maintain the integrity of your security level (make sure that your file permissions don't leave themselves wide open to attack/access). COPS does not detect bugs in software that could cause security problems (such as tftpd or sendmail), and it does not correct any errors that are found.

NOTE: Many vendors of UNIX are now bundling a security status checker with the OS, usually under the nomenclature of a "C2" or "trusted system." You may still find that this package has more features than your canned package. Compare them.

Availability: COPS can be retrieved via anonymous FTP from cert.org in ~/pub/tools/cops.

C2. Chkacct. Chkacct is a COPS for the ordinary user. This tool is made available to the users to run, or it is run for them once per day. It will do an integrity check on the status of files in their own account and then mail them the results (such as "Dear user: Your .rhosts file is unsafe"). This package can help make your users more aware of security controls and raise their level of participation in the program.

Availability: Chkacct is distributed with the COPS package, however you may retrieve the most recent version of Chkacct via anonymous FTP from cc.perdue.edu in ~/pub/chkacctv1.1.tar.Z.

C3. CRACK. Crack helps the security administrator identify weak passwords by checking for various weaknesses and attempting to decrypt them. If Crack can figure out your password, then you must choose a better password. It is very likely that a determined intruder will be able to get the password too (using similar techniques, or the Crack program itself, since it is publicly available).

Availability: Crack is available via anonymous FTP from cert.org in ~/pub/tools/crack/crack_4.1-tar.Z.

C4. SHADOW. The shadow password suite of programs replaces the normal password control mechanisms on your system to remove the encrypted password from the publicly readable file /etc/passwd and hides them in a place that only this program has permission to read. It consists of optional, configurable components, provides password aging to force users to change their passwords once in awhile, adds enhanced syslog logging, and can allow users to set passwords up to a length of sixteen characters.

NOTE: Many vendors of UNIX are now bundling a shadow password suite with the OS, usually under the nomenclature of a "C2" or "trusted system." You may still find that this package has more features than your canned package. Compare them.

Availability: Shadow is available from USENET archives which store the comp.sources.misc newsgroup. Distribution is permitted for all non-commercial purposes. For more information contact the author, John F. Haugh III (jfh@rpp386.cactus.org).

C5. PASSWD+. Passwd+ is a proactive password checker that replaces /bin/passwd on your system. It is rule-based and easily configurable. It prevents users from selecting a weak password so that programs like "CRACK" can't guess it, and it provides enhanced syslog logging.

NOTE: Many vendors of UNIX are now bundling a proactive password checker with the OS, usually under the nomenclature of a "C2" or "trusted system." You may still find that this package has more features than your canned package. Compare them.

Availability: Passwd+ is available via anonymous FTP from dartmouth.edu in ~/pub/passwd+tar.Z.

C6. AUDIT. Audit is a policy-driven security checker for a heterogeneous environment. It is fully configurable so that you can set up Audit to exactly match your site's security policy. This program functionally does what COPS is intended to do, but does not hard-code your policy decisions for you the way that COPS does.

NOTE: Many vendors of UNIX are now bundling an auditing subsystem with the OS, usually under the nomenclature of a "C2" or "trusted system." You may still find that this package has more features than your canned package. Compare them. One particular subject to note is that most (IMHO) vendors auditing subsystems only collect and regurgitate tons of raw data, with no guidance and assistance for using that information. They leave that up to you. The Audit and/or Swatch tools are probably better.

Availability: The final version of Audit will eventually be posted to USENET. However, the beta release will only be made available on a limited basis, to larger, heterogeneous sites. If your interested in participating in the beta test, send e-mail to the auther, Bjorn Satdeva (bjorn@sysadmin.com).

C7. MIRO. Miro is a suite of tools for specifying and checking security contraints (like COPS and Audit), including a couple programming languages. It is general because it is not tied to any particular OS, and it is flexible because security administrators express site policies via a formal specification language. It is easy to extend or modify a policy by simply augmenting or changing the specification of the current policy.

Availability: Miro is the product of a large research project, and to understand it you need more than the paragraph I've written above. For more information about the Miro project send e-mail to (miro@cs.cmu.edu), there is even a video available. The authors Ph.D thesis, as well as the sources for the Miro tools, are available via anonymous FTP from ftp.cs.cmu.edu. When you connect there, type "cd /afs/cs/project/miro/ftp" and "get ftp-instructions"; this will explain how to get the thesis and/or software.

SECTION D - Additional Security Enhancements

The tools described in sections {A...C} above, are what I consider part of a "base" set of tools and functional requirements for general security administration. The tools and methods described in this section are additional measures that can be combined with or added to your overall security program at any of the other levels {A...C}.

D1. One-time Password ("Key Card"). Since reusable passwords can be captured and used/reused by intruders, consider a "one-time password" scheme. One-time passwords can be implemented using software-only solutions or software/hardware solutions, and there are several commercial products available. The following is an example of what CERT uses. Each user is assigned a "Digital Pathways" key-card (approximately $60 per user). When you enter your PIN code, it supplies a password that is good only one time. The only other piece to this, is software that replace the login shell on your "firewall" server.

Availability: The source-code for this shell is public-domain and is available via anonymous FTP from cert.org. For additional information about this, send e-mail to (cert@cert.org).

D2. Privacy Enhanced Mail (PEM). PEM is a RSA-based encryption scheme that encrypts sensitive information, but more than that it checks for message integrity and non-repudiation of origin, so that the originator cannot deny having sent the message. PEM is actually a protocol that is designed to allow use of symmetric (private-key) and asymmetric (public-key) cryptography methods. In this example, Trusted Information Systems, Inc. (TIS) has implemented a PEM package using the public-key technique together with the Rand MH Message Handling System (version 6.7.2). TIS/PEM libraries can be adapted for implementation of non-mail applications as well.

Availability: TIS/PEM is a commercially available product, for additional information send e-mail to (pem-info@tis.com).

D3. Kerberos. Kerberos is a DES-based encryption scheme that encrypts sensitive information, such as passwords, sent via the network from client software to the server daemon process. The network services will automatically make requests to the Kerberos server for permission "tickets." You will need to have the source to your client/server programs so that you can use the Kerberos libraries to build new applications. Since Kerberos tickets are cached locally in /tmp, if there is more than one user on a given workstation, then a possibility for a collision exists. Kerberos also relies upon the system time to operate, therefore it should be enhanced in the future to include a secure time server (timed is not appropriate). There are two versions of Kerberos, one for OSF ported by HP, and one BSD-based developed by the author.

Availability: Kerberos is distributed via anonymous FTP from athena-dist.mit.edu in ~/pub/kerberos or ~/pub/kerberos5.

D4. Private-Key Certificates. This is not really a product, but rather a design proposal that is an alternative method to PEM for adding network security to applications such as mail. Simply put, it uses the public-key style of implementation with private-key cryptography. It can be adapted to different types of applications and it is boilerplate so that you can essentially plug-in any encryption algorithm. This is designed so that public-key protocols no longer have to rely on public-key encryption.

Availability: Unknown. For more information, contact Don Davis, at Geer Zolot Assoc., Boston, MA (formerly of Project Athena at MIT). His paper "Network Security via Private-Key Certificates" better describes this techique.

D5. Multi-Level Security (MLS). I don't have any particular product in mind here, rather to point out that after you've done everything else (above) to make your network secure, then MLS will probably be one of your next logical steps. That doesn't mean you have to wait until you've done everything else before implementing MLS, it's just (IMHO) that you would be wasting your time to go to the Nth degree before covering the fundamentals. Many UNIX vendors are now shipping or preparing to ship a MLS version. An example that immediately comes to mind is AT&T System V/Release 4/MLS (SVR4/MLS). I don't have any firsthand experience with this to offer you, but I do have some secondhand comments. Basically, you can buy MLS as part of your OS, but all you've really bought are the tools to build your MLS security program with. From what I have heard, it takes considerable effort in software development to develop a set of programs and procedures in order to use MLS UNIX effectively as an MLS environment. If anyone has any further information to dispute or expand this claim I would be very interested in hearing it.

D6. File Encryption. Users should get into the habit of encrypting sensitive files whenever they are stored in a public place or transmitted via public communication circuits. File encryption isn't bulletproof, but it is better than clear text for sensitive information. The UNIX crypt utility is the least secure of these tools, since it can be broken using well-known decryption techniques. The UNIX des utility (available in US only) is more secure. It has not been known to be broken,

however DoD does not sanction its use for transmitting classified material. A new UNIX tool PGP 2.0 is available (uses RSA encryption), however there may be licensing issues to be concerned with.

D7. Secure Programming Methods. Programmers can assist in the effort of security by reducing the chance that a potential intruder can exploit a hole or bug that is coded into locally developed software. There is probably a lot that can be said about this, and their are probably many books on the subject somewhere. But, here are some common recommendations. (A) Never create a SETUID shell script. There are well-known techniques used by intruders to gain access to a shell program that is running as root. (B) List the complete file name, including the full path in any system() or popen() call. (C) Since there is no reason for users to have read access to a SETUID file (or any exectuble for that matter), set permissions to 4711 (SETUID) or 711 (Non-SETUID).

D8. Counterintelligence. To extend your security program to seek out, identify, and locate intruders; you may want to modify some of the security tools (especially those proxy service daemons and event-driven auditors) to trace intruders back to their source, and otherwise maintain logs of data on intrusion attempts. This information can prove vital in taking an offensive stance against security break-in's and can help prosecute offenders.

D9. Other Possibilities. Depending on your requirements you might look into specialized solutions such as Compartmented Mode Workstations (CMW), end-to-end Data Link Encryption, and TEMPEST. The NCSC (Rainbow Series) and ITSEC specifications can help you define what level of need you have for security and help lead you to additional types of solutions.

SECTION E - Security Policy

Everything discussed in sections {A...D} involve specificgs you can do, tools and techniques to implement, to address a particular area or "hole" in security. Your SECURITY POLICY is what ties all of that together into a cohesive and effective SECURITY PROGRAM. There are many diverse issues to consider when formulating your policy, which alone is one of the biggest reasons why you must have one:

- What are the functional requirements of your network?
- How secure do you need to be? What needs to be protected?
- How will you handle incident reporting and prosecution?
- What does the law require you to do? What about privacy? Since break-ins often occur via multiple hops on computers throughout the US and the rest of the world, you will need to consider a variation of federal, state, local, as well as foreign laws.
- Make security a dedicated and deliberate effort.
- User training and security awareness.
- What is considered acceptable use for users? Do the users understand what it is they are permitted to do and what it is they are not permitted to do?
- What is considered acceptable use for system administration staff? Is using Crack to test passwords okay? Is giving friends outside the organization accounts okay?
- Maintain a working relationship with the Computer Emergency Response Team (CERT) at Carnegie Mellon University (CMU) and your Forum of Incident Response and Security Teams (FIRST) regional representative "CERT" team.
- PLUS a myriad of different issues too numerous to go into in a summary paper.

By answering these questions you determine what packages and methods in sections {A...D} that you want to implement, and in what ways you want to modify or configure them. "A security policy is a formal specification of the rules by which people are given access to a computer and its resources." (and to extend that to say...a network and its resources). Whatever tools you install to help you maintain the security of your network and monitor it, they must be configured

to implement YOUR POLICY, or else they are not doing the whole job that needs to be done. Therefore, you must first have a POLICY.

For additional help in the area of policy development, contact cert@cert.org, and they can probably lead you to useful documentation on the subject and guide you to your FIRST regional CERT team representative. A good starting point is Request For Comments (RFC) 1244 "Site Security Handbook" (96 pages), which is available via anonymous FTP from numerous RFC archive sites (for example: nic.ddn.mil).

SUMMARY OF AVAILABILITY

Section	Name	Availability
A1	Router	Cisco, Wellfleet, Proteon
B1	Tcp_wrapper	cert.org:/pub/tools/tcp_wrappers
B2	Socks	e-mail to koblas@sgi.com
B3	Kernel_wrap	eecs.nwu.edu:/pub/securelib
B4	Swatch	sierra.stanford.edu:/pub/sources
B5	CAP	e-mail to thompsond@orvb.saic.com
B6	Mail Gateway	NOT APPLICABLE
B7	Tty_wrapper	NOT APPLICABLE
C1	COPS	cert.org:/pub/tools/cops
C2	Chkacct	cc.perdue.edu:/pub/chkacctv1.1.tar.Z
C3	Crack	cert.org:/pub/tools/crack/crack_4.1-tar.Z
C4	Shadow	comp.sources.misc (jfh@rpp386.cactus.org).
C5	Passwd+	dartmouth.edu:/pub/passwd+tar.Z
C6	Audit	e-mail to bjorn@sysadmin.com
C7	Miro	e-mail to miro@cs.cmu.edu
D1	Key-card	e-mail to cert@cert.org
D2	TIS/PEM	e-mail to pem-info@tis.com
D3	Kerberos	athena-dist.mit.edu:/pub/kerberos5
D4	Private-key	contact Don Davis, at Geer Zolot Assoc.
D5	MLS	contact your UNIX vendor
D6	File encrypt	contact your UNIX vendor
D7	Programming	NOT APPLICABLE
D8	Counter-Intel	NOT APPLICABLE
D9	Other Poss	research and contact various vendors
E*	Policy	RFC 1244 and cert@cert.org

Additional Sources of Information

Subscribe to the following mailing lists:

cert-advisory-request@cert.org
cert-tools-request@cert.org

Read the following USENET newsgroups:

comp.security.announce (echos the CERT advisory mailing list)
comp.security.misc
alt.security (frequently dissolves into "flame wars")
comp.risks
comp.virus (almost exclusively for discussing PC and MAC viruses)

Copy files from the CERT Usenet Clipping Archive via anonymous FTP from cert.org

CERT Contact Information:

Emergencies: +1 412 268-7090
FAX: +1 412 268-6989
E-mail: cert@cert.org

U.S. Mail: CERT Coordination Center
Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213-3890, USA

USENIX Papers are available directly from USENIX:
The USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710, USA

READ: The SYMPOSIUM PROCEEDINGS of the USENIX Third UNIX Security Symposium (September 14-16, 1992). It is 330+ pages, containing papers describing in better detail a lot of the packages and security techniques I briefly described in this paper.

----------NOTICE---DISCLAIMER----------

The contents of this paper do not necessarily reflect the opinions of my employer or anyone else that I know. Nothing in this paper should be construed as a product endorsement. No warranty is expressed or implied. Any comments? Please send me e-mail.