

# Unix Flaws and Vulnerabilities

Everyday, all over the world computer networks and hosts are broken into. The level of sophistication varies widely; while it is generally believed that most break-ins succeed due to weak passwords, there are still a large number of intrusions that use more advanced techniques to break in which are harder to detect.

UNIX as it comes “out of the box” is full of vulnerabilities, flaws and inconsistencies, largely due to the fact that UNIX was not designed with security in mind.

In this project we will show, using UNIX, that even seemingly harmless network services can become vulnerable tools in the search of a weak point of a system, even though these services are operating exactly as they are intended to.

Although there are many weak points within the UNIX system, we will emphasise what we feel to be the major ones.

## 1. Buffer Overflow

### 1.1 Introduction

A buffer overflow is where an input data goes beyond the range that the receiving buffer is allocated and this excess is not dealt with gracefully.

If the buffer is on the stack an arbitrary piece of code could be executed following the creation of a fake stack frame.

This is exploited by intruders being able to force the program to execute arbitrary commands, these “exploit” commands will run with the UserID of the original program, which is usually root.

#### Why does it occur?

Programmers sometimes fail to observe correct programming procedures:

- **Failure to check boundary conditions**  
e.g. letting 25 characters be inputted where the input buffer has only room for 20
- **Failure to check return codes of functions/procedures**  
If a buffer overflow occurs in a function, failure to check returned values could lead to problems.
- **Pointer problems / wrap around**  
Where a pointer can be caused to wraparound an array and give unexpected results.

## 1.2 Vulnerabilities

- **Gain of an unauthorised privilege (e.g. root)**
- Program failure
- Data loss

### 1.2.1 Some Silicon Graphics Irix commands <sup>1</sup>

Command name	Details	System affect
Pset	Used to display and modify information concerning the use of processor sets in the current system. The pset command is used on multi-processor systems to restrict the execution of different classes of jobs.	Local users may gain the privileges of group sys. These privileges may then be used to gain root privileges.
Df	Used to display statistics about the amount of used and free disc space on file systems.	Local users may gain root privileges
Eject	Used to eject a removable media device, such as floppy, CDROM, or tape. If the floppy or CDROM is mounted, eject will first try to unmount it.	
Login/scheme	Used at the beginning of each terminal session that allows users to identify themselves to the session. Under current versions of IRIX, this functionality is supplied by the program /usr/lib/iaf/scheme. (The login program is a symbolic link to /usr/lib/iaf/scheme).  Although this vulnerability has been verified only under IRIX 6.2, it is believed to affect other versions of IRIX, including IRIX 5.x.	
Ordist	Used to maintain identical copies of files over multiple hosts. It preserves the owner, group, mode and mtime of a file if possible.	

### 1.2.2 Sendmail<sup>2</sup>

Sendmail version 8 contains support for MIME, the central idea behind MIME is defined in RFC 1341:

*"... designed to provide facilities to include multiple objects in a single message, to represent body text in character sets other than US-ASCII, to represent formatted multi-font text messages, to represent non-textual material such as images and audio fragments"*

The support in sendmail version 8 includes data translations in which a message's body is either stripped to 7-bit ASCII, achieved by forcing the 8th bit to be off, or 8-bit MIME, achieved by leaving the 8th bit as is.

Sendmail can be configured for either of these translations.

With the release of sendmail version 8.8.3, a security vulnerability was introduced that allows remote users to execute arbitrary commands on the local system with root privileges.

By sending a carefully crafted email message to a system running a vulnerable version of sendmail, intruders may be able to cause a buffer overflow and force sendmail to execute arbitrary commands with root privileges. Those commands are running on the same system where the vulnerable sendmail is running.

In most cases, the MIME conversion of an email is done on final delivery; that is, to the local mailbox or a program. Therefore, this vulnerability may be exploited on systems *despite firewalls and other network boundary protective measures*.

Remote users can gain root privileges on a machine running sendmail versions 8.8.3 or 8.8.4 that does 7-to-8 bit conversion. They do not need access to an account on the system to exploit the vulnerability.

### 1.2.3 Xwindows<sup>3</sup>

The X Window System™ provides the base technology for developing graphical user interfaces. X contains various methods for manipulating basic elements of the GUI which are loaded and drawn on the user's screen for sending user interactions back to the application. Security implications arise because at its lowest level X Windows is a communication protocol called X Protocol which uses a client-server model of network communication. The user works directly from the X Server (opposite to the usual convention) running applications (X Clients) on the same machine or remote hosts. The most common client is Xterm – a terminal emulator. This section describes flaws relating to buffer overflows in X Windows, which are frequently found in the program libraries and any clients linked to these libraries. The libraries we refer to here are Xaw \* – the widgets library, X11 and Xt but there are others.

\* The Athena widget set is a library package that provides a two-dimensional set of user interface tools sufficient to build a wide variety of applications. This layer extends the basic abstractions provided by X and provides the next layer of functionality primarily by supplying a cohesive set of sample widgets. There is no standard widget set.

## Problem descriptions

- 1.) Current problems exist in both the Xterm program and the Xaw library that cause buffer overflows allowing users to gain unauthorised privileges. They occur with the processing of data related in particular, to the inputMethod and preeditType resources (both for Xterm and Xaw) and the \*Keymap resources (Xterm).

The resources and the releases affected by the Xterm vulnerability are:

Release	InputMethod	PreeditType	*Keymap
X11R3	NO	NO	YES
X11R4	NO	NO	YES
X11R5	NO	NO	YES
X11R6	NO	NO	YES
X11R6.1	YES	YES	YES
X11R6.2	YES	YES	YES
X11R6.3	YES	YES	YES
X11R6.4	YES	YES	YES

The resources and the releases affected by the Xaw library vulnerabilities are:

Release	InputMethod	PreeditType
X11R6	YES	YES
X11R6.1	YES	YES
X11R6.2	YES	YES
X11R6.3	YES	YES
X11R6.4	YES	YES

- 2.) X11R6.1 and earlier contains buffer overflows in any setuid or setgid program linked to library X11.
- 3.) All Redhat Linux installations contain buffer overflows in any setuid or setgid program linked to LibXt.<sup>4</sup>
- 4.) Many Sun OS and Solaris OS contain buffer overflows in any setuid or setgid program linked to LibX11 and LibXt.<sup>5</sup>
- 5.) One X11R6 Xserver command accepts any value for a runtime argument regardless of it's length. This causes a buffer overflow which local users can exploit to gain extra privileges or even root privileges when Xserver is setuid root.<sup>6</sup>

## 1.3 Solutions

Vendors occasionally fix solutions such as these, more often though a workaround is issued.

One example of a typical workaround for the above is to prevent the exploitation of these vulnerability by removing the setuid or setgid and non-root execute permissions of the effected programs.

In many situations it is normal practice to disallow some of these programs to non-root users.

E.g. df: To prevent the exploitation of the vulnerability you should remove setuid permissions from the df program immediately. As df will no longer work for non-root users, we recommend removing the execute permissions for them.

```
# ls -l /sbin/df
-r-sr-xr-x 1 root sys 23136 Nov 22 1994 /sbin/df
# chmod 500 /sbin/df
# ls -l /sbin/df
-r-x----- 1 root sys 23136 Nov 22 1994 /sbin/df
```

## 2 SETUID

### 2.1 Introduction

e.g.: `chmod u+s file-name.`  
This lets the program 'file-name' execute with the owner's privilege level.

This feature allows non-root users to temporarily become privilege users in order to access files that require privilege.

The most common example is the '*passwd*' program, which a user executes whenever he/she needs to change the password.

#### Why does it occur?

Most older versions of Unix implementations allow normal user to create shell programs that are setuid or setgid with the owner set to be root, i.e. any user can become super user (privilege user) by executing these shell programs.

Programs with setuid set that provide shell escape to let users run other programs are subjected to attack if the special privilege was not removed. Any program that a user invokes at this stage (before the setuid is reset) is running with root privilege!

Refer to the **sendmail** (Version older than 8.8.8) example of setgid problem.

## 2.2 Vulnerabilities

### 2.2.1 X Windows

- 1) In versions of the X11 program Xterm local users can use the logfile facility to create or modify files on the system enabling unauthorised access including root. The problem only occurs in systems where Xterm is installed with setuid or setgid privileges and when the logfile facility is enabled.<sup>7</sup>
- 2) Seyon is a setuid program providing serial port communications for X Windows. It is vulnerable only to users with local access who can exploit the problem to gain local or remote root privileges.<sup>8</sup>
- 3) The X11 audio mpeg player, when installed setuid root, creates playlist files making root the owner. This can enable any user to overwrite or delete /etc/shadow for example.<sup>9</sup>

### 2.2.2 Sendmail

- 1) When sendmail delivers mail to a program listed in a .forward or :include file, that program is run with the group permissions possessed by the user who owns that .forward or :include: file. The file's owner attribute is used to initialise the list of group permissions that are in force when the program is run.<sup>10</sup>

It is possible for users to get group permissions they should not have by linking to a file that is owned by someone else, but on which they have GroupWrite permissions. By changing that file, users can obtain the group permissions of the owner of that file.

Exploitation is possible if the attacked user has a file that is group writable by the attacker on the same file system as either (1) the attacker's home directory or (2) an :include: file that is referenced directly from the aliases file and is in a directory writable by the attacker. The first attack only works against root. This attack does not give users root "owner" permissions, but does give them access to the groups that list root in /etc/group.

**Impact:** A local attacker can gain the group permissions of another user.

- 2) On some systems, setuid and setgid scripts (scripts written in the Cshell, Bourne shell, and Perl, for example, with the set user or group ID permissions enabled) are insecure due to a race condition in the kernel. For those systems, Perl version 4 and 5 attempts to work around this vulnerability with a special program named suidperl, also known as Sperl.<sup>11</sup>

Suidperl attempts to emulate the set-user-ID feature of the kernel. Depending on whether the script is set-user-ID, set-group-ID, or both, suidperl achieves this emulation by first changing its effective user or group ID to that of the original perl script. Suidperl then reads and

executes the script as that effective user or group. To do these user and group ID changes correctly, `suidperl` must be installed as set-user-ID root.

On systems that support set-user-ID and set-group-ID, `suidperl` does not properly relinquish its root when changing its effective user and group IDs.

**Impact:** On a system that has the `suidperl` or `sperl` program installed and that supports saved set-user-ID and saved set-group-ID, any one with access to an account on the system can gain root access.

## 2.3 Solutions

If using an older version of Unix system, apply a patch or upgrade to a newer version. If this option is not available, check through the File system for any file that has `setuid` and verify that they are the files that are supposed to have `setuid/setgid` set.

For the `sendmail` problems there are published workarounds involving the `UnsafeGroupWrite` option in the configuration file and a `fixperl` script that replaces the `setuid` and `sperl` programs with a wrapper.

# 3 Passwords <sup>12</sup>

## 3.1 Introduction

Unix uses the *Password* file (`/etc/passwd`) to keep track of users on the system. Each line of the password file typically contains the Username/User-account, UserID, Usergroup, Real user name or comment, home directory and the default login shell. The fields are separated by ‘:’. The `/etc/passwd` file is readable by everyone and writable by *root* only. Current Unix versions store passwords in encrypted form, mostly in a separate file called *shadow* file in `/etc/shadow`. The shadow file is set to be read by root only.

## 3.2 Vulnerabilities

The *crypt* function is considered secure, as there is no known technique to decrypt the stored password back into plaintext. As a result, the only way to defeat the Unix password is by using brute force (e.g. dictionary) attacks, the most popular case being the Internet Worm attack in 1988.

## 3.3 Solutions

To prevent an attacker from using dictionary attacks by pre-encrypting an entire dictionary, Morris and Thompson modified the way passwords are encrypted using the crypt function and the way it is stored in the `/etc/shadow` file.

Whenever a user enters a password, a *salt value* - a 12 bit number, which is based on the time of day converted into a 2 character string - is stored together with the encrypted password. This means that the *same* password can encrypt into 4096 ( $2^{12}$ ) different ways. This makes the job of building the database of all encrypted password slightly more difficult.

## 4 Remote Shell Access

### 4.1 Introduction

Before internetworking became commonplace, protecting a system from unauthorised access simply meant locking the machine in a room by itself. Now that machines are connected by networks security is much more complex. This section describes the tools and methods available to make your UNIX network as secure as possible.

#### Trusted Hosts

One of the most convenient features of the Berkeley (and Sun) UNIX networking software is the concept of “trusted” hosts. The software allows the specification of other hosts (and possibly users) who are to be considered trusted - remote logins and remote command executions from these hosts will be permitted *without* requiring the user to enter a *password*. This is very convenient, because users do not have to type their password every time they use the network.

### 4.2 Vulnerabilities

Unfortunately, for the same reason, the concept of a trusted host is extremely insecure. The Internet worm made extensive use of the trusted host concept to spread itself throughout the network. Many sites that had already disallowed trusted hosts did fairly well against the worm compared with those sites that did allow trusted hosts. Even though it is a security hole, there are some valid uses for the trusted host concept.

#### 4.2.1 Host.equiv & .rhost

- 1) The file `/etc/hosts.equiv` can be used by the system administrator to indicate trusted hosts (typically by IP address or host name). Each trusted host is listed in the file, if a user attempts to log in (using `rlogin`) or execute a command (using `rsh`) remotely from one of the systems listed in `hosts.equiv`, and that user has an account on the local system with the same login name, access is permitted without requiring a password. Therefore it is vulnerable to IP spoofing.

Non-local hosts (including hosts at remote sites of the same organisation) should never be trusted. Also, if there are any machines at your organisation that are installed in “public” areas (e.g. terminal rooms) as opposed to private offices, you should not trust these hosts. Further precautions are listed in the Unix Computer Checklist AUCERT.<sup>13</sup>

- 2) The `.rhosts` file is similar in concept and format to the `hosts.equiv` file, but allows trusted access only to specific host-user combinations, rather than to hosts in general. Each user may create a `.rhosts` file in his home directory, and allow access to his account without a password. Most people use this mechanism to allow trusted access between accounts they have on systems owned by different organisations who do not trust each other’s hosts in `hosts.equiv`. Unfortunately, this file presents a major security problem. While `hosts.equiv` is under the system administrator’s control and can be managed effectively, a user may create a `.rhosts` file granting access to whomever he/she chooses, without the system administrator’s knowledge.

The only secure way to manage `.rhosts` files is to completely disallow them on the system. The system administrator should check the system often for violations of this policy. One possible exception to this rule is the ‘root’ account; a `.rhosts` file may be necessary to allow network backups and the like to be completed.

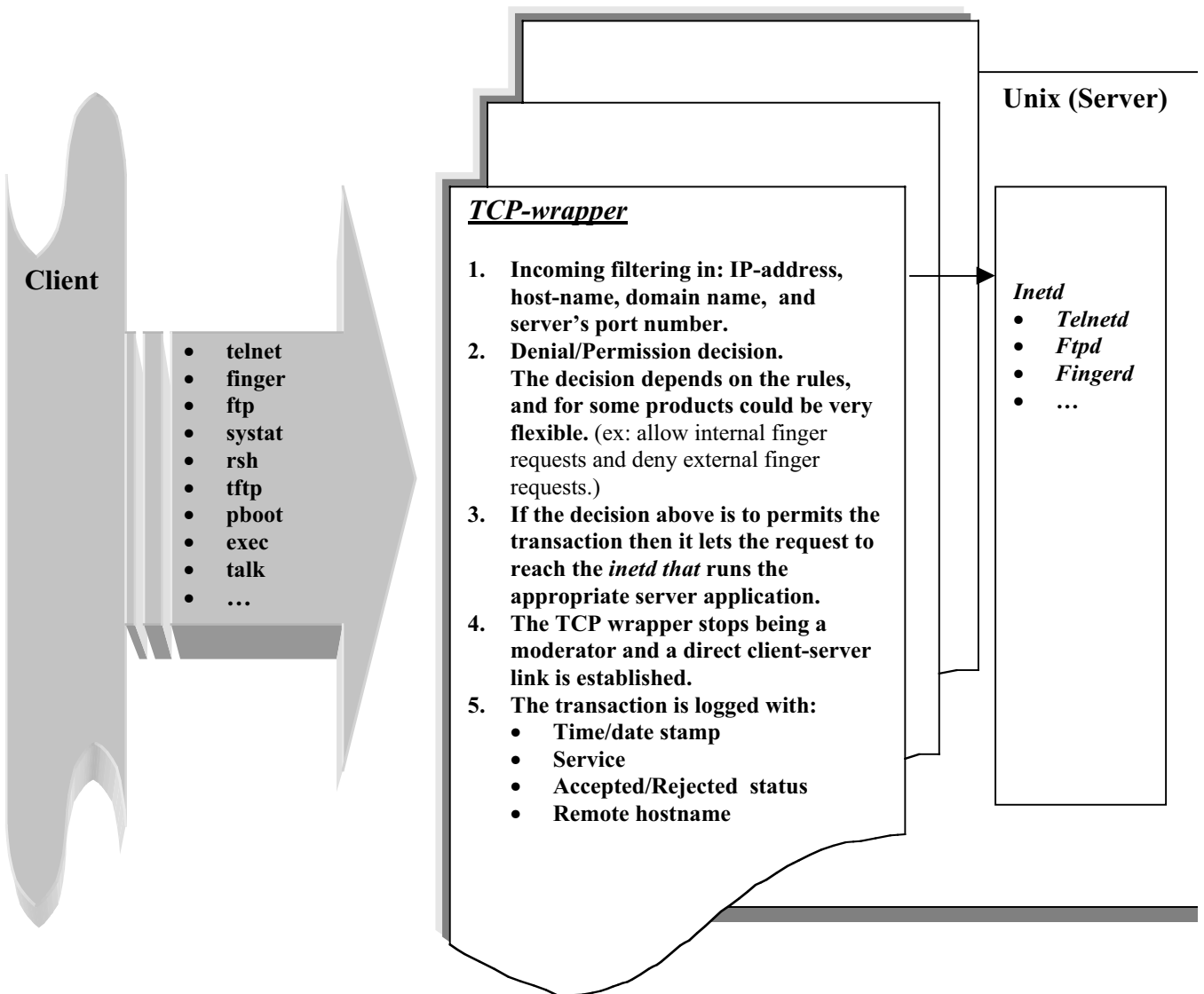
#### 4.2.2 X Windows<sup>14</sup>

- 1) Xhost is analogous to the `host.equiv` concept in that this command lists trusted host names that have access to the Xserver. It is simple to implement but has drawbacks:
  - X Server is open to everyone if your configuration includes “x host +”and the off the shelf configuration is insecure
  - It’s simplicity can be a drawback if you have many users on a single host (very common for machines running X Server) then any user with access to the host has access to the Server.
  - Authentication is often based on IP address or Host name making it vulnerable to IP spoofing attacks.
  - Some servers read a small packet from the client to determine if they are in the xhost list, if the packet is not sent the server may halt operation indefinitely or until timed out. This causes a denial of service attack.
- 2) Token Authentication is analogous to the `.rhosts` file concept, it allows access only to specific host-user (i.e. X Client) combinations based on the X Client providing a Magic Cookie or 56 bit DES key and 64 bit random authenticator. This enhanced security is transparent to the user:
  - Cookies are transmitted over the network with out encryption so they can be intercepted.
  - Cookies and random authenticators are stored in the users `.Xauthority` file which may be read by an unauthorised user.
  - X Display manager (Xdm) is one way of creating the magic cookie. In X11R6 or earlier this cookie was guessable as it used a poor pseudo random number generator (later versions use DES to generate cryptographically secure cookies).

- Using token authentication is time consuming for both user and administrator and requires a good understanding of the X Client – Server model
- 3) Xterm's write access feature means that SendEvents (key and button events generated artificially) may be accepted from the X Server. This allows any user to send commands or keystroke sequences to a remote terminal and opens up communication from sources other than the user who initiated it.
  - 4) Xterm's read access feature provides a Secure Keyboard option, which means that all keyboard events are sent exclusively to the xterm window. This prevents critical keyboard events, like password entry, being captured by other clients.

## 5 TCP-Wrappers

TCP-wrapper is a program that acts like a “security-receptionist” in an internet-connected UNIX-system. It provides it with the capability of monitoring and filtering the incoming Internet-related traffic on a UNIX-server. The first TCP-wrapper was created in Eindhoven University of Technology (Netherlands), in about 1990, after continuous hacking attacks in their Internet connected UNIX server.



### Notes/ Problems

- TCP-wrapper does not take any (direct) action to client or server applications.
- Usually works only for TCP-services started from *inetd*.
- Does not work for RPC-based services and programs that start-up a daemon that is running out *inetd*.
- Obviously a very useful tool in a security armour for monitoring and controlling incoming traffic, BUT nowadays modern firewalls have more intelligent and flexible packet filtering and monitoring methods.
- If TCP-wrappers are not set up properly one can impersonate a trusted machine and bypass them.
- One can not filter the outgoing packets.
- When one gains *root* then he/she is able to modify the TCP-wrapper files.
- Wrappers can be used to connect users to proxy service rather than the actual service

## 6 Appendix

### 6.1 The Internet Worm Exploit<sup>15</sup>

A key attack of the Worm program involved attempts to discover user passwords. It was able to determine success because the encrypted password of each user was in a publicly readable file. This allows an attacker to encrypt lists of possible passwords and then compare them against the actual passwords without passing through any system function. In effect, the security of the passwords is provided in large part by the prohibitive effort of trying all combinations of letters. Unfortunately, as machines get faster, the cost of such attempts decreases.

Dividing the task among multiple processors further reduces the time needed to decrypt a password. It is currently feasible to use a supercomputer to precalculate all probable 8 passwords and store them on optical media. Although not (currently) portable, this scheme would allow someone with the appropriate resources access to any account for which they could read the password file and then consult their database of pre\_encrypted passwords. As the density of storage media increases, this problem will only get more severe. A clear approach to reducing the risk of such attacks, and an approach that has already been taken in some variants of UNIX, would be to have a shadow password file. The encrypted passwords are saved in a file that is readable only by the system administrators, and a privileged call performs password encryption and comparisons with an appropriate delay (.5 to 1 second, for instance). This would prevent any attempt to “fish” for passwords. Additionally, a threshold could be included to check for repeated password attempts from the same process, resulting in some form of alarm being raised. Shadow password files should be used in combination with encryption rather than in place of such techniques, however, or one problem is simply replaced by a different one; the combination of the two methods is stronger than either one alone.

Another way to strengthen the password mechanism would be to change the utility that sets user passwords. The utility currently makes minimal attempt to ensure that new passwords are nontrivial to guess. The program could be strengthened in such a way that it would reject any choice of a word currently in the on-line dictionary or based on the account name.

---

**References:**

- <sup>1</sup> [http://www.cert.org/advisories/CA-97.21.sgi\\_buffer\\_overflow.html](http://www.cert.org/advisories/CA-97.21.sgi_buffer_overflow.html)
- <sup>2</sup> <http://www.cert.org/advisories/CA-97.05.sendmail.html>
- <sup>3</sup> [http://www.cert.org/vul\\_notes/VN-98.01.Xfree86.html](http://www.cert.org/vul_notes/VN-98.01.Xfree86.html)  
[http://www.cert.org/ftp/cert\\_bulletins/VB-98.04.xterm.Xaw](http://www.cert.org/ftp/cert_bulletins/VB-98.04.xterm.Xaw)  
<http://www.opengroup.org/tech/desktop/x/>  
<http://sunsite.doc.ic.ac.uk/>
- <sup>4</sup> <http://www.ciac.org/ciac/bulletins/h-67.shtml>
- <sup>5</sup> <http://www.ciac.org/ciac/bulletins/h-100.shtml>  
<http://www.ciac.org/ciac/bulletins/h-108.shtml>
- <sup>6</sup> <http://oliver.efri.hr/~crv/security/bugs/mUNIXes/x11r6.html>
- <sup>7</sup> <http://www.ciac.org/ciac/bulletins/e-04.shtml>
- <sup>8</sup> <http://www.ciac.org/ciac/bulletins/i-089.shtml>
- <sup>9</sup> <http://oliver.efri.hr/~crv/security/bugs/mUNIXes/x11amp.html>
- <sup>10</sup> [http://www.cert.org/advisories/CA-96.25.sendmail\\_groups.html](http://www.cert.org/advisories/CA-96.25.sendmail_groups.html)
- <sup>11</sup> [http://www.cert.org/advisories/CA-96.12.suidperl\\_vul.html](http://www.cert.org/advisories/CA-96.12.suidperl_vul.html)
- <sup>12</sup> Paper by Robert Morris and Ken Thompson --- Bell Laboratories
- <sup>13</sup> [http://www.deter.com/unix/papers/unix\\_security\\_checklist.txt](http://www.deter.com/unix/papers/unix_security_checklist.txt)
- <sup>14</sup> [http://ciac.llnl.gov/ciac/documents/CIAC-2316\\_Securing\\_X\\_Windows.pdf](http://ciac.llnl.gov/ciac/documents/CIAC-2316_Securing_X_Windows.pdf)  
Simson Garfinkel, Gene Spafford: "Practical Unix & Internet Security" (O'Reilly&Associates, Inc)
- <sup>15</sup> <http://www.cne.gmu.edu/modules/acmpkp/security/texts/INTWORM.PDF>