

The NetBSD Update System

Alistair Crooks, The NetBSD Project

9th April 2004

Abstract

This paper explains the needs for a binary patch and update system, and explains the background and implementation of NetBSD-update, a binary update facility for NetBSD. The implementation is then analysed, and some lessons drawn for others who may be interested in implementing their own binary update system using the NetBSD pkgsrc tools, which are available for many operating systems and environments already.

The NetBSD Binary Update System

Unix, Linux and the BSD operating systems have traditionally been distributed in source format, and users and administrators have had a long tradition of compiling utilities and applications from source. Over time, however, vendors have moved towards a binary-only distribution mechanism, removing various parts of the system in the process, such as the C compiler, and other necessary tools. It is only over the last decade that the rise of Linux and the BSD operating systems have placed the emphasis back on source code, and even then, most versions of the operating systems are installed from a binary distribution.

This paper describes the NetBSD update system, which has been implemented on and for the NetBSD operating system <http://www.netbsd.org/> using the packaging tools from the NetBSD Packages Collection (pkgsrc) <http://www.pkgsrc.org/>.

Other vendors and operating systems have binary update facilities in place - their existence is not the

driving force behind the use of pkgsrc or NetBSD - rather, this is a description of a facility which is used in NetBSD and which can be used on any other operating system to augment the standard facilities which are in place.

Driving Forces for a Binary Patch and Update System

It is now common to find firewalls in large and small organisations, preventing malign access, and protecting the organisation from intrusion and other attacks. It would not be prudent to have a C compiler installed on such a machine - its use should be that of a gatekeeper, as a bouncer with an attitude, keeping anything suspicious out, and not allowing anyone who does manage to penetrate the defences to use any tools to break further into the infrastructure.

In addition to these instances, it is very unusual to find users (as opposed to administrators, or people who work in the industry) who would know what to do with a source distribution. Email to various mailing lists proves this point - to the majority of users out there, the computer is a tool, not a thing of beauty.

It is also common to find vulnerabilities in operating systems, libraries, and utilities which have already been deployed. To fill such holes, a patching system needs to be used - the vulnerable code is replaced by code which is not vulnerable. This must be done by means of updating the binaries.

Embedded systems must also be examined in light of the vulnerabilities found in MTAs, network time servers, IP filtering software, operating system software, and anything else which is included as part of

the embedded system, and which can be used to facilitate an attack on an organisation or individual - the whole infrastructure is only as strong as the weakest link in the defences, and one breach renders the rest of those defences useless.

It is not feasible to expect that all the vulnerabilities can be found and protected ahead of time - new ones are being found every day - and so there must be a mechanism available by which new binaries can be used to overwrite vulnerable ones. This paper examines all the issues involved in the design and deployment of such a system.

It can take some time for an advisory to be received, investigated, researched, fixed, and then publicised - the binary update facility can be viewed as a more reliable form of communication of the exploit or vulnerability than reading the Bugtraq or Full Disclosure mailing lists.

Some vendors wish users and administrators to pay for the added service which a binary update facility provides. Whilst the NetBSD binary update system is not intended to prevent vendors charging for this service, some people are unhappy and unwilling to pay for a binary update system.

Related Systems

The one immediate piece of software to which everyone refers when talking about binary updates is the Windows Update Facility <http://v4.windowsupdate.microsoft.com/en/default.asp>. It has three separate facilities - update check (for new upgrades), update download, and update installation, and Windows XP computers can be set to perform the check, the check and download, and all three parts automatically. Windows Update is based around a web front end - all the user has to do is to specify (the first time Windows Update is run) how they would like their system to check for updates - and everything is very easy from an end-user point of view.

Debian Linux <http://security.debian.org/> has a security update feature which uses its apt system to provide binary patches for Debian Linux systems.

The FreeBSD project <http://www.freebsd.org/> are working on their own binary update system.

The RedHat network <http://www.redhat.com/> provides a commercial service for RedHat Linux systems.

Sun Microsystems <http://www.sun.com/> provides a number of binary updates via its SunSolve facility for its Solaris operating environment.

The NetBSD Packages Collection <http://www.pkgsrc.org/> has an "audit-packages" package which is used to notify the user when packages which are installed on the host system have been found to have security exploits or vulnerabilities. A central list is maintained by the security-officer and other developers, and stored on one of the NetBSD project servers. The user is encouraged to run the "download-vulnerability-list" as part of the periodical jobs on a host system by means of the crontab(8) facility. This script will download the list of vulnerabilities from the central server. Output from the "download-vulnerability-list" script looks is reproduced in the Figure download-vulnerability-file output.

A separate script, called "audit-packages" simply runs through each of the entries in the list of vulnerabilities, and checks against each of the vulnerabilities for an installed package on the system with a version number that is vulnerable. If there is, then a warning message will be printed, with relevant information, as shown in Figure audit-packages.

The vulnerabilities file is split into 3 columns. Each line of the file describes a new vulnerability. Comments start with the '#' character. The first column is the vulnerable package versions; the second column is the type of vulnerability, and the final column is a URL related to the vulnerability.

Some typical entries from the vulnerabilities list are shown in the Figure pkg-vulnerabilities file.

The audit-packages package works very effectively in practice, having ironed out a few loose ends in its implementation. From the start, the vulnerability list has been advertised that it will only grow - old vulnerabilities will be retained "just in case". This has proved difficult to implement - occasionally URLs will be found to be out of date, or better ones identified. Once a user with a shorter username fixed two transposed characters in a comment, thereby causing the size of the vulnerabilities list to decrease. In time, it was found better to have an embedded SHA1 di-

gest, which is used to ascertain correct transmission of the vulnerabilities list, and that no truncation has occurred in transit, as well as ensuring that the file has not been modified in any way.

NetBSD also has fine-grained “system packages”, which can be used as an alternative to the traditional BSD sets method of installing or upgrading systems. System packages are usually made up of three sub-packages:

- the binaries,
- the manual pages,
- and any example or support files.

The NetBSD Update System does not use any of the facilities of system packages - on non-NetBSD systems, it is likely that system packages would clash with traditional methods of installing and updating systems, and so a separate mechanism was used to implement the NetBSD Update System.

The Design of the NetBSD Update System

The audit-packages package, and the experience gained from it, proved to be a major influence in the design of the NetBSD update system. By using a single list of vulnerable packages which was downloaded, significant useful information could be retrieved very simply and in a relatively secure manner.

The utility and ease-of-use of the Windows Update system proved to be the inspiration for the actual operation of the system. In addition to the ease of use, the three stages of protection in the Microsoft utility were considered to be the correct approach. In some countries, users must opt-in to any service which holds information, and the binary update facility was designed with this in mind.

Security and integrity were two more considerations - security in the transit of information which could be potentially damaging to a system if the update were modified in any way during transit. For this reason, digital signatures of the update itself are considered absolutely essential.

The author has some experience with the packaging tools as used in the NetBSD packages collection, and certain facilities used in these tools would be needed in the binary update facility itself. Using version numbers which could be compared rationally with each other, and using the tools themselves to add, backup and delete files, directories and other file system entries are essential in providing a useful utility which does not try to do any more than it needs to do.

Implementation of the NetBSD Update System

The implementation of the NetBSD update system is done as in the Windows and Debian updates - the binary updates can be set to inform of new updates, to inform and download, and to inform, download and install the updates automatically.

The NetBSD packages collection tools are used to perform the binary update work - these tools have been enhanced from the original tools to add such features as “Dewey decimal” number comparison, digitally-signed package checks and addition, and package replacement in place. There is also a facility for a binary package to be specified by URL on the command line, downloaded automatically, and added via `pkg_add(1)`, but that facility is not used by the NetBSD binary update system - instead, `ftp(1)` is used to download the binary package, and `pkg_add(1)` is used in “local” mode. This is because we prefer the binary packages which are being downloaded to be protected by a digital signature, and that mode of binary package addition is, unfortunately, not yet available when specifying a URL on the command line.

For a long time, the NetBSD `pkgsrc` system has included a package called “`pkg_install`” - it consists of the latest version of the `pkg_install` tools. This was derived from the original FreeBSD `pkg_install(1)` tools (and is also included in the base NetBSD operating system), but has been considerably enhanced, including specifically such features as

- the recognition of packages by means of relative version numbers

- the addition of a callout to gpg and pgp to authenticate the provenance of a binary package by means of a digital signature related to the binary package. This had to be implemented as a callout as gpg is distributed under the GPL, and it is not desirable to have any part of this system distributed under a more restrictive or onerous licence than the BSD licence. The implication of this is obvious - the BSD operating systems need a BSD-licensed utility for privacy, compatible with gpg and pgp. One example of an addition of a signed binary package is shown in Figure Addition of a signed binary package.

The recognition of packages by comparative version numbers is essential for any binary update system. For example, it is imperative that we can tell whether an installed package is vulnerable to an exploit. There are a number of ways this could be done

- circulate digests of files that are vulnerable, but that is not practical in an environment such as the BSD operating systems where most packages are built by the user, and binary packages, whilst still being used by many people, are not as popular as the “build it for myself” attitude. The author suffers from this affliction as well, and does not consider it a weakness, more the signs of knowing exactly what is on any given system at any one time.
- by hardcoding the package names which are vulnerable, but this does not scale well, or would even be practical
- by using a system package name and version number (but, as we have already seen, this is not portable to operating systems other than NetBSD)

The NetBSD packages collection has also long had a number of “meta-packages” - these are packages which include no files or directories of their own, but which require pre-requisites of other packages, and thereby ensure that all of the parts can be manipulated as one whole package. Examples of meta-packages are:

- edesktop

- gnome
- gnome2
- gnustep
- kde3
- netbsd-www
- etc

The First Implementation

As it was first implemented, the NetBSD binary update system was actually a fairly simple shell script which used off-the-shelf tools to perform its work. Some thought had been placed into making it into a standalone compiled program, since that is the vehicle which would most readily benefit embedded systems vendors, and that is left as future work.

The system itself consisted of two parts:

- an `$osys-$osversion-$architecture-update` meta-package
- the individual updates in the form of binary packages

The netbsd-update facility first checked the directory on the NetBSD ftp server to see if there was a more recent meta-package than the one currently installed on the host machine. If there was, this was notified to the user.

Then, when the time came to download the new updates, the new meta-package was downloaded, as well as any binary packages which were necessary. Whilst it was usual for an update to include one binary package for that architecture and version, more may be needed. It’s also possible that a binary package may itself have been updated, in which case the newer version of that was downloaded, along with the updated meta-package.

The operating system version and architecture are used in the package name to ensure that the correct binaries are downloaded (the architecture), and that the binary package was linked with the correct system libraries (operating system version).

When the time came to install the binary updates, the normal `pkg_add(1)` tool was used, working from a local copy of the package. The “-s” switch was used to ensure that the digital signature (in the form of a separate ASCII-armoured file) for the binary package verified the update’s authenticity (much as a signed RPM package is used in Linux).

An overview of the NetBSD Update Facility

Building on the prototype building blocks outlined above, the NetBSD Update facility has to perform the following steps:

1. ascertains the current operating system name and version
2. downloads a list of the binary packages available for the operating system and version of the operating system
3. evaluate whether any binary packages are needed - the same process which is undertaken by `audit-packages` takes place here. The list of available binary packages is processed, using `pkg_info(1)` to work out whether the binary package is installed on the system or if it needs to be.
4. if the user has specified that they just want to be informed of the existence of an update (which is not recommended), then this takes place by means of mail to root, and no further actions take place.
5. the necessary available packages are downloaded from the ftp server, along with their digital signatures.
6. if the user has specified that they just want the available binary packages downloaded, then they are informed by mail (to root) that this has happened, and no further actions take place
7. the files which will be overwritten (if any) by extracting files from the available binary packages are preserved by means of a simple `tar(1)` command, using the file list from the available binary

packages, and made into a binary package of its own. This also facilitates rollback, should an unsuccessful binary package addition take place, or the new behaviour with the update in place is not working as intended

8. the binary package is fed to the `pkg_add(1)` utility, which first checks the digital signature on the binary package and verifies it. It is possible to specify that any signatories which match a regular expression will be allowed - for example “security-officer@netbsd.org”, or “.*@pkgsrc.org”
9. the binary package is added to the system by means of `pkg_add(1)`.

Consequences of these steps:

- the user is able to specify in advance whether they want the binary package notification, automatic download, or applied to the system. This is the same system as the Windows Update facility, and has been found by numerous people from small users to large enterprises to work very well.
- digital signatures protect the user by assuring them of the provenance of the binary package. Without this assurance, it would be risky or downright dangerous to apply third-party binary packages to any system, far less one which may be in production use.
- the ability to rollback from an update is simply the deletion of the downloaded binary package, and the re-application of the preserved binary package.
- people must be running a GENERIC kernel for the update system to be of any use. This is not really a problem in reality, since very few people tailor or customise their kernel configurations (usually, the only people that do that are developers)

Feedback from Initial Usage

- The user interface was originally clunky, and needed to be made smoother

- Problems with Firewall-1 and interaction with NetBSD's ftp server mean that http may well be a better vehicle for binary update downloading
- The digital signature and automatic updates do not co-exist very well together. It is anticipated that the pkg_install tools will be modified to use a configuration file for trusted signatures.
- Operating systems which are not NetBSD were not catered for very well. Having said that, there was nothing to preclude their use on other operating systems, although duplication of updates from SunSolve, for example, are not useful in themselves; what is useful, however, is a coherent set of packages for a given release of Solaris, for relevant architectures and machines. It has been the author's experience in the past, when managing networks of Solaris machines, that Sun provide the updates, but do not group them together particularly well, and that the numerical nature of the updates names are not the easiest to remember or inform others.
- The quality of the binary updates is only as good as the port-masters for NetBSD who have to build the binary packages.
- Making meta packages for each individual architecture is onerous and time-consuming. Some knowledge of package creation is needed.
- a digital signature of the vulnerabilities file can be added in a trivial manner, to protect the integrity of the information
- using a simple vulnerabilities file means that there is no confusion of different files per architecture or operating system, no duplication of information

The refined binary update facility now performs the following steps:

- download the vulnerabilities file from the NetBSD ftp server
- ensure its integrity and provenance by calling out to gpg to verify the digital signature
- if there is a vulnerability on the host system, this will be flagged by a down-level update package (from a previous vulnerability fix), or by the update package not existing on the host machine.
- if the update package is to be downloaded, use ftp(1) to download the binary package from the NetBSD ftp server. At the same time, the digital signature will be downloaded. Typically, they will both be downloaded together as one entity, to try to minimise any spoofing attacks, or man in the middle attacks, since ftp is hardly the most secure of protocols
- if the update package is to be installed automatically, if there are any files to be preserved (so that rollback can take place), then the package tools are used to create a binary package of the preserved files. The list of files to be preserved will be included with the update package itself as part of its meta data.
- if the update package is to be installed, use pkg_add(1) with its "-s gpg" argument to verify the package's integrity, and to add it
- the root user will be informed of all of the above steps by email. It is believed by the author that it is better to err on the side of too much information, especially where far-reaching changes to

Iterative Development

Learning from the experience gained from the prototype implementation, it was decided that some changes would benefit everyone:

- rather than having a separate meta-package for each operating system, and operating system version, and architecture, a single vulnerabilities file was used, mirroring the vulnerabilities file in the "audit-packages" package
- the vulnerabilities file can also use an embedded digest, as a further check that correct transmission has taken place - although this is not as useful or protective as:

the system could be taking place. Update packages can have their information displayed using the normal `pkg_info(1)` interface.

Conclusions

The binary update system is a useful piece of work, which has shown positive benefits.

- The ability to provide better security warnings and recovery is an enormous benefit
- even if a fix is not available, the binary update system can be used to publicise an exploit in the wild, thereby making the administrator's job easier (assuming that information about the exploit is available). Even being aware that a vulnerability has been found in a widely-used piece of software is a valuable service
- the use of digital signatures to protect and disseminate the information has proved to be of immense benefit

There are certainly areas for development and improvement, but the cross-platform and cross-environment nature of the NetBSD packages collection can aid other operating systems and environments as well as simply the BSD operating systems.

Future Work

The NetBSD packages collection (`pkgsrc`) runs on the following operating systems:

- AIX
- BSDOS
- Darwin (Mac OS X)
- FreeBSD
- Interix (Microsoft's Services for Unix)
- IRIX
- Linux
- NetBSD

- OpenBSD
- SunOS (Solaris)

whilst more - HP/UX, the Hurd, Digital Unix - are planned. We would like to extend the binary update facility to run on as many of those operating systems as possible, as the benefits are measurable. It is certainly the case that duplication of work would be inadvisable, but the binary update facility offers a real benefit to end users.

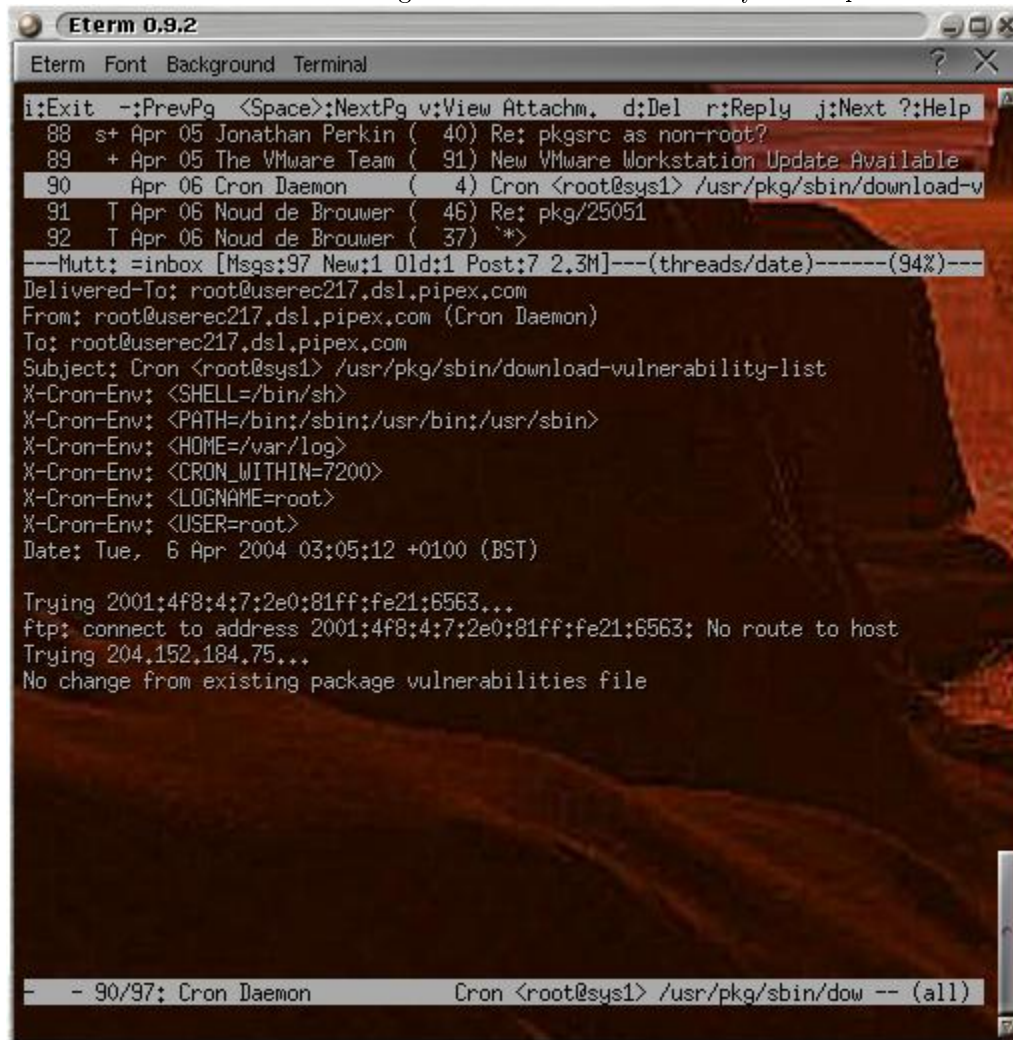
It would be beneficial to investigate using a protocol that is more secure than ftp to download the vulnerabilities file, update packages, and digital signatures. Some form of anonymous secsh could be useful for this project.

A BSD-licensed gpg utility, perhaps based on openssl, would mean that callouts to gpg could be avoided, and that the whole binary update system would be licensed with a less restrictive licence than the GPL.

References

1. The NetBSD Operating System - <http://www.netbsd.org/>
2. The NetBSD Packages Collection - <http://www.pkgsrc.org/>
3. The Microsoft Windows Update Facility - <http://v4.windowsupdate.microsoft.com/en/default.asp>
4. Debian Linux Security Update - <http://security.debian.org/>
5. The FreeBSD Project - <http://www.freebsd.org/>
6. The Red Hat Network - <http://www.redhat.com/>
7. Sun's SunSolve database - <http://sunsolve.sun.com/>

Figure 1: download-vulnerability-file output

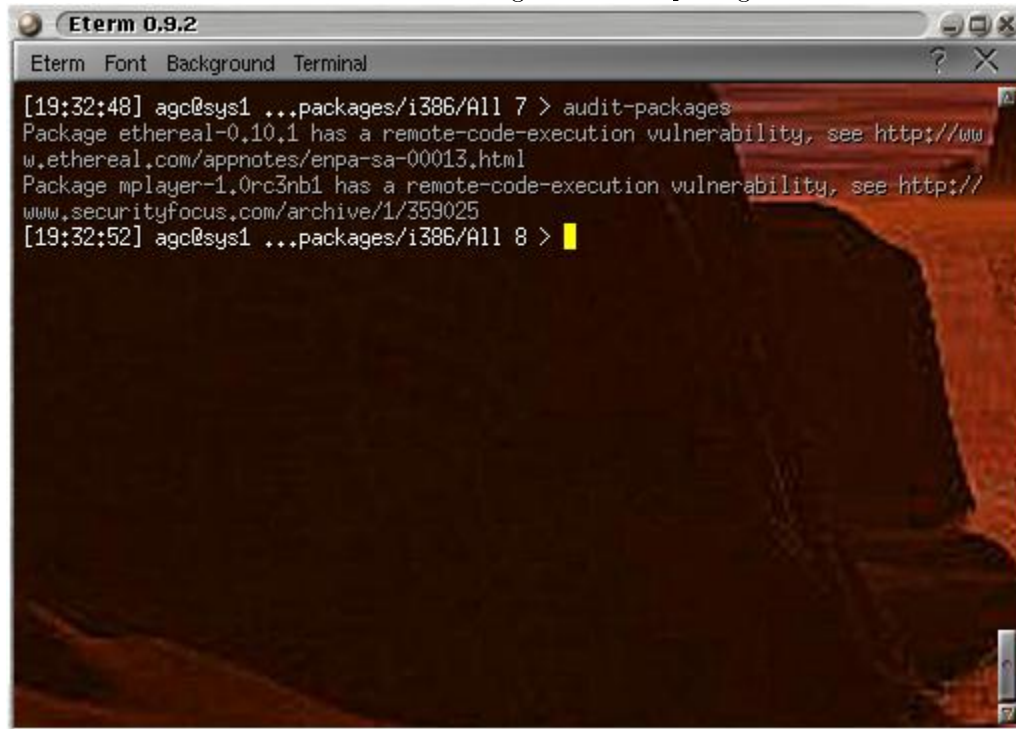


```
Eterm 0.9.2
Eterm Font Background Terminal
i:Exit -:PrevPg <Space>;NextPg v;View Attachm. d:Del r:Reply j:Next ?:Help
88 s+ Apr 05 Jonathan Perkin ( 40) Re: pkgsrc as non-root?
89 + Apr 05 The VMware Team ( 91) New VMware Workstation Update Available
90 Apr 06 Cron Daemon ( 4) Cron <root@sys1> /usr/pkg/sbin/download-v
91 T Apr 06 Noud de Brouwer ( 46) Re: pkg/25051
92 T Apr 06 Noud de Brouwer ( 37) `*>
---Mutt: =inbox [Msgs:97 New:1 Old:1 Post:7 2.3M]---(threads/date)----- (94%)---
Delivered-To: root@userrec217.dsl.pipex.com
From: root@userrec217.dsl.pipex.com (Cron Daemon)
To: root@userrec217.dsl.pipex.com
Subject: Cron <root@sys1> /usr/pkg/sbin/download-vulnerability-list
X-Cron-Env: <SHELL=/bin/sh>
X-Cron-Env: <PATH=/bin:/sbin:/usr/bin:/usr/sbin>
X-Cron-Env: <HOME=/var/log>
X-Cron-Env: <CRON_WITHIN=7200>
X-Cron-Env: <LOGNAME=root>
X-Cron-Env: <USER=root>
Date: Tue, 6 Apr 2004 03:05:12 +0100 (BST)

Trying 2001:4f8:4:7:2e0:81ff:fe21:6563...
ftp: connect to address 2001:4f8:4:7:2e0:81ff:fe21:6563; No route to host
Trying 204.152.184.75...
No change from existing package vulnerabilities file

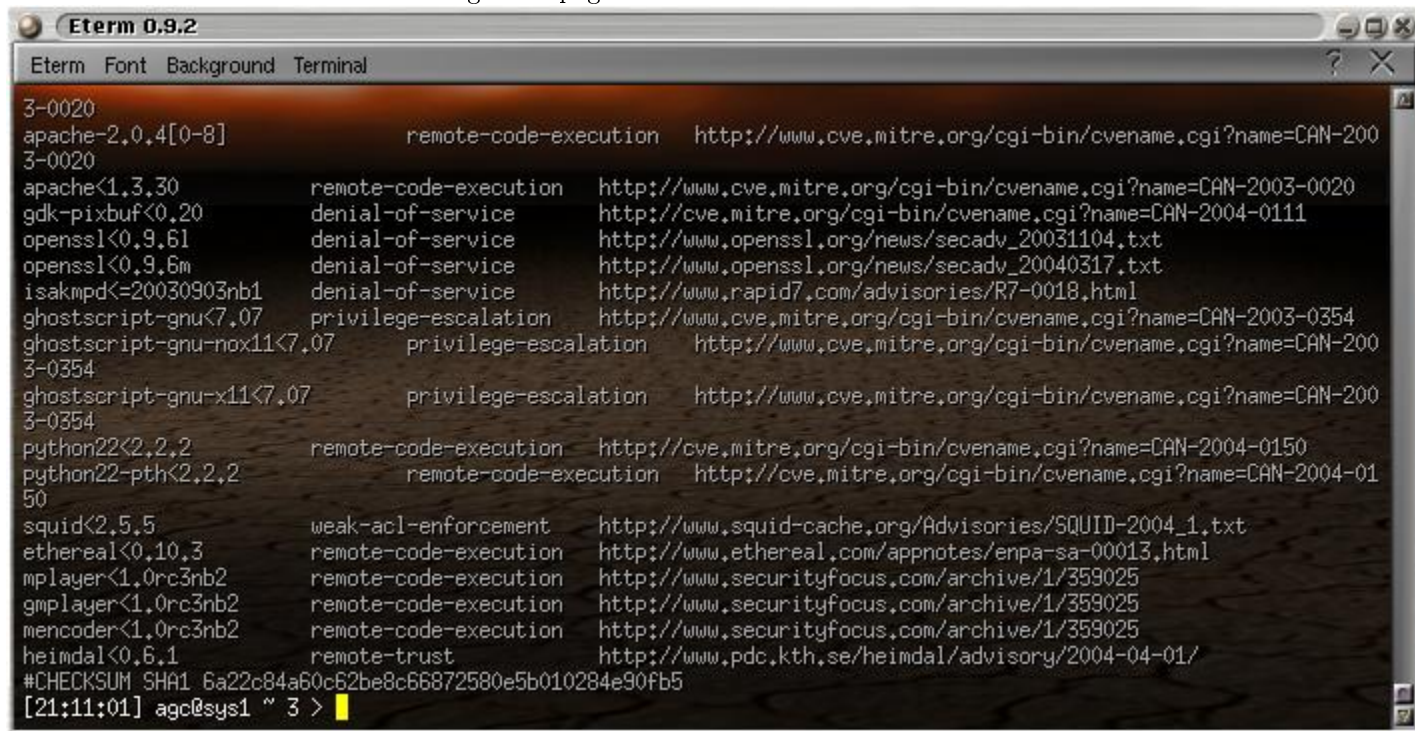
- 90/97: Cron Daemon Cron <root@sys1> /usr/pkg/sbin/dow -- (all)
```


Figure 2: audit-packages



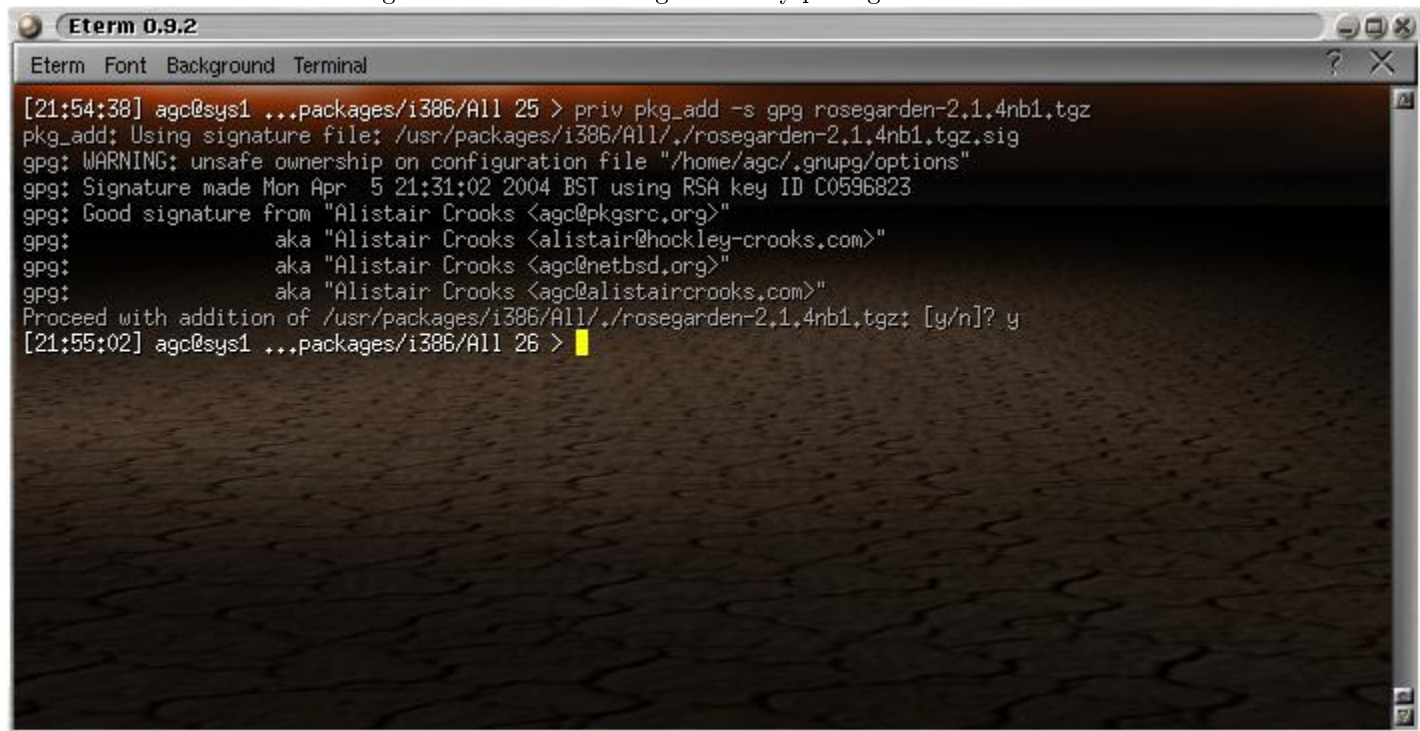
```
Eterm 0.9.2
Eterm Font Background Terminal
[19:32:48] agc@sys1 ...packages/i386/All 7 > audit-packages
Package ethereal-0.10.1 has a remote-code-execution vulnerability, see http://www.ethereal.com/appnotes/enpa-sa-00013.html
Package mplayer-1.0rc3nb1 has a remote-code-execution vulnerability, see http://www.securityfocus.com/archive/1/359025
[19:32:52] agc@sys1 ...packages/i386/All 8 > █
```

Figure 3: pkg-vulnerabilities file



```
Eterm 0.9.2
Eterm Font Background Terminal
3-0020
apache-2.0.4[0-8]          remote-code-execution  http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0020
3-0020
apache<1.3.30             remote-code-execution  http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0020
gdk-pixbuf<0.20           denial-of-service      http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0111
openssl<0.9.6l           denial-of-service      http://www.openssl.org/news/secadv_20031104.txt
openssl<0.9.6m           denial-of-service      http://www.openssl.org/news/secadv_20040317.txt
isakmpd<=20030903nb1     denial-of-service      http://www.rapid7.com/advisories/R7-0018.html
ghostscript-gnu<7.07     privilege-escalation   http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0354
ghostscript-gnu-nox11<7.07  privilege-escalation   http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0354
ghostscript-gnu-x11<7.07  privilege-escalation   http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0354
python22<2.2.2           remote-code-execution  http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0150
python22-pth<2.2.2       remote-code-execution  http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0150
squid<2.5.5              weak-acl-enforcement   http://www.squid-cache.org/Advisories/SQUID-2004_1.txt
ethereal<0.10.3          remote-code-execution  http://www.ethereal.com/appnotes/enpa-sa-00013.html
mplayer<1.0rc3nb2        remote-code-execution  http://www.securityfocus.com/archive/1/359025
gmplayer<1.0rc3nb2       remote-code-execution  http://www.securityfocus.com/archive/1/359025
mencoder<1.0rc3nb2       remote-code-execution  http://www.securityfocus.com/archive/1/359025
heimdal<0.6.1            remote-trust            http://www.pdc.kth.se/heimdal/advisory/2004-04-01/
#CHECKSUM SHA1 6a22c84a60c62be8c66872580e5b010284e90fb5
[21:11:01] agc@sys1 ~ 3 >
```

Figure 4: Addition of a signed binary package



```
Eterm 0.9.2
Eterm Font Background Terminal
[21:54:38] agc@sys1 ...packages/i386/All 25 > priv pkg_add -s gpg rosegarden-2,1,4nb1.tgz
pkg_add: Using signature file: /usr/packages/i386/All/./rosegarden-2,1,4nb1.tgz.sig
gpg: WARNING: unsafe ownership on configuration file "/home/agc/.gnupg/options"
gpg: Signature made Mon Apr  5 21:31:02 2004 BST using RSA key ID C0596823
gpg: Good signature from "Alistair Crooks <agc@pkgsrc.org>"
gpg:      aka "Alistair Crooks <alistair@hockley-crooks.com>"
gpg:      aka "Alistair Crooks <agc@netbsd.org>"
gpg:      aka "Alistair Crooks <agc@alistaircrooks.com>"
Proceed with addition of /usr/packages/i386/All/./rosegarden-2,1,4nb1.tgz: [y/n]? y
[21:55:02] agc@sys1 ...packages/i386/All 26 > █
```