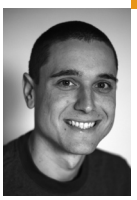
JUSTIN SAMUEL AND JUSTIN CAPPOS

package managers still vulnerable: how to protect your systems



Justin Samuel is an undergraduate student at the University of Arizona. He is the author of the RequestPolicy Firefox extension, a tool for increasing the privacy and security of browsing.

jsamuel@cs.arizona.edu

Justin Cappos is a Post Doc at the University of Washington. His research interests revolve around building large distributed systems and gaining experience from their practical use.

justinc@cs.washington.edu

SERIOUS VULNERABILITIES HAVE BEEN discovered in all Linux package managers. Although most major package managers and distributions have begun addressing these issues, some of the most popular are not fully protecting their users. We'll look at known vulnerabilities in package managers, what is being done to fix them, and how you can protect your systems even if you're using a vulnerable package manager.

By package managers, we're talking about the tools we all use to update the software on our systems, such as APT, YUM, and YaST. Even if you don't invoke your package manager directly but instead do so through a graphical interface or a scheduled job, the same vulnerabilities exist.

The vulnerabilities we discovered require that an attacker be able to respond to a client when the package manager is downloading files. Unfortunately, we also found that it is extremely easy, in many cases, for attackers to position themselves to do this. By becoming a public mirror for a distribution's repositories, attackers can often target as many clients as they have bandwidth to support [1].

Here we'll look at the most common package managers and the distributions using them, which are, generally:

APT:	used by Debian and Ubuntu
APT-RPM:	used by ALT Linux and PCLinuxOS
Pacman:	used by Arch Linux
Portage:	used by Gentoo
Slaktool:	used by Slackware
Stork:	used for research on PlanetLab
URPMI:	used by Mandriva
YaST:	used by openSUSE and SUSE Linux Enterprise
YUM:	used by Fedora, Red Hat Enterprise, and CentOS

In this article we will focus on Linux package management systems. With BSD systems, a common approach for updates is the distribution of the Ports collection [2]. The Ports collection is often distributed insecurely, although the portsnap tool for FreeBSD, which we are currently investigating, may be a secure alternative [3].

How Package Managers Work

Package managers do the job of installing new or updated software on our systems. To be able to do this, they run as root. Thus, what they do and

the software they install affect the security of the entire system. If a package manager can be tricked by an attacker into installing a malicious package the attacker has created (or even just an old package that has known vulnerabilities), the attacker can compromise the system. It's therefore critical that package managers be secure.

To be secure, they must install the software we intended them to install. If we didn't want a package installed (e.g., a vulnerable or malicious package), it shouldn't be installed. If we did want a package installed (e.g., an update that fixes a security flaw), that package should be installed. Also, of course, a malicious party shouldn't be able to cause our package manager to crash our system while it is running.

For all package managers, the basic process they follow is similar. First, they download information about available packages from a remote repository. They then use this information to decide which package to install. The repository is usually an HTTP or FTP server. The information downloaded (called *metadata*) gives details about the packages that are available on the repository. These details include information about the versions of each available package, any additional packages they require in order to work properly, what functionality they provide, their cryptographic hashes, their file sizes, and so on.

In practice, a common approach employed by most package managers is to start off each time they run by downloading a single file from the repository. This file, the *root metadata*, is a small file that describes the contents and layout of the repository, usually giving the names and hashes of other files on the repository that actually contain the detailed information the package manager needs. The package manager determines whether the files listed in the root metadata are different from the ones it last downloaded; if so, it downloads these new files from the repository and verifies that what it received has the same hashes as were listed in the root metadata.

After downloading all of the metadata it needs, the package manager uses the information to decide whether there is anything that should be installed. If the package manager was being run for the purpose of updating the system, it looks at all of the packages currently installed and checks whether the metadata describes newer versions (higher version numbers) of those same packages. If the package manager is being used to install a new package on the system, it looks for the highest version number of that package listed in the metadata.

When deciding whether a package can be installed, the package manager makes sure that either the other software a package requires is already installed or, if not, that this software can also be obtained from the repository (that is, that it is available according to the metadata). This process of *dependency resolution* continues until there are no more dependencies to resolve, there is some form of conflict, or some dependencies cannot be resolved. The package manager only continues if it can resolve all dependencies.

Next, the package manager downloads the individual packages it wants to install and then installs them. At this point, if your package manager has been tricked by an attacker, you're in big trouble.

Traditional Security in Package Managers

Some package managers don't make any pretense of being secure in the first place (such as Slaktool on Slackware and Pacman on Arch Linux). We

strongly recommend against using package managers that don't try to be secure.

Of course, most popular package managers do, in fact, try to perform the update and installation process described here securely. Until recently, most of them considered themselves completely secure.

As you may know or have already guessed, their security mechanisms are based on the use of cryptographic signatures. The security differences in current package managers largely come down to what is actually signed. The options are either signatures on the root metadata, omn the packages themselves, or, in some cases, on metadata that describes the packages.

Signatures on the root metadata mean that the first file the package management client downloads has a signature the client can check. By checking the root metadata file's signature and then verifying that the secure hashes of each file downloaded thereafter match the expected hashes, the signature's authority extends to the ultimately downloaded packages. The package managers that use this model include APT, APT-RPM, and YaST.

Another common place for the signature is on each individual package. In this model, the package manager has no signatures to check until it gets to the point where it downloads the actual packages it intends to install. It then checks the package signatures before installation. Package managers that use signatures on individual packages include YUM and URPMI.

Finally, signatures can be placed on files that directly contain the metadata of the packages. This approach is used by Portage and Stork, although they accomplish this in somewhat different ways [1].

The Vulnerabilities

The vulnerabilities that were discovered fall into three main categories: replay/freeze attacks, metadata manipulation attacks, and denial-of-service (DoS) attacks.

REPLAY AND FREEZE ATTACKS

The replay attacks that package managers are vulnerable to are the same replay attacks that cryptanalysts and security-minded people have long been aware of. The attack in this case involves a malicious party responding to a package manager's request for signed metadata (the information about packages available on a repository) with an old signed file. The attacker does not need to compromise the signing key to do this.

The problem basically comes down to the fact that, with the way package managers currently work, once a file is signed and thus trusted by clients, it is always trusted, even after vulnerabilities are discovered in packages that were once considered safe. This, of course, will always happen (which is a large part of why we use package managers: so we can get updates to patch our systems once vulnerabilities are discovered).

So, a replay attack allows an attacker to respond to a client's request for repository metadata with old metadata that lists packages the attacker knows how to exploit. This could even be metadata much older than that which the client has already seen. Unfortunately, with the way all package managers have been written, clients aren't bothered if yesterday they retrieved metadata that is dated from last week but today they retrieved metadata that is dated from a year ago.

A freeze attack is similar to a replay attack. In fact, from a cryptanalytic point of view, it's actually the same thing. However, it's worth giving it a different name to ensure it gets the attention it deserves. This is because solving all facets of the replay attack problem isn't as simple as making sure that clients never accept metadata that is older than metadata they have already seen. As an attacker can keep giving the client a single version of the metadata starting at one point in time (that is, "freezing" the metadata), the attacker can prevent the client from knowing about new metadata and thus new packages that are available that fix known vulnerabilities.

Therefore, securing package managers against replay and freeze attacks must involve limiting how long any signed metadata is considered valid. The problem is that no package managers currently do this (though a few are now working on this, as we'll see in a moment).

METADATA MANIPULATION ATTACKS

The replay attacks discussed so far are useful for attackers who want to compromise systems whose package managers use signed metadata. However, if a package manager does not used signed metadata (such as with YUM or URPMI), attacks don't have to bother with replaying old metadata. Instead, attackers can just make up their own metadata!

What's the point of attackers making up their own metadata that they serve to clients? There are two main things attackers can do. First, they can mix-and-match the versions of packages that are listed. Second, they can trick clients into thinking that packages have different dependencies and provide different functionality than they really do.

In mixing-and-matching vulnerable package versions by listing them in the same metadata given to a client, attackers make it more likely that, whatever new package a client installs, it is installing a version with a known vulnerability. At least with replay attacks, the attacker has to choose a certain snapshot of historical packages to show the client. If two packages were vulnerable at different times, the attacker can't use a replay attack to make both vulnerable versions available to a client. However, when a package manager doesn't sign metadata, the attacker isn't limited to a single point in time.

As mentioned, the lack of signed metadata allows attackers to lie to clients about what each package provides and requires. By lying to clients about this information, an attacker can significantly increase the chances of a client installing a vulnerable package. For example, if package *foo* has a vulnerability the attacker knows how to exploit, the attacker can provide metadata that says every package depends on package *foo,* in order to ensure that the client installs it when installing any other package.

There are other bad things an attacker can do when package managers don't use signed metadata. The point, though, is that if your package manager does not sign metadata, your systems are at risk. The solution here is for clients to require signed metadata. A package manager should at least sign the root metadata. Depending on the design of the package manager, it may also use signatures on package metadata.

ENDLESS DATA ATTACKS

The last of the categories of attacks on package managers that were made known comprises forms of DoS attacks. These are not intellectually challenging concepts, but they have been universally overlooked and their ramifications are quite serious.

The attack involves a malicious party responding to a client request, be it for metadata or for a package, with an endless stream of data. Depending on the package manager and the specifics of the stream of data (e.g., whether the endless data began after http headers were ended or before), the effects can vary even within the same package manager. The possible effects include filling up the partition where the package manager saves downloaded files or exhausting memory.

Obviously, in either case, bad things can happen. If the file system where the package manager is storing downloaded files is somewhere that other important data is stored (it's usually under "/var/"), that other important data can be corrupted. Likely candidates for corrupted data are databases, mail, and log files. If memory is exhausted, the system slows to a crawl (an attacker could, for example, slow down the stream of endless data before memory is completely used and the OS kills the package manager's process).

Mirrors: The Easy Way to Exploit Clients

How does an attacker go about exploiting these vulnerabilities? Fundamentally, the attacker just needs to be able to respond to the client's requests for metadata and packages. This ends up being easier than it sounds.

For most noncommercial Linux distributions, the repositories from which package managers download files are not actually run by the distributions themselves. Instead, a wide variety of volunteer individuals, companies, and organizations donate a portion of their own server space and bandwidth by acting as repository mirrors. These mirrors of the main repository do nothing other than provide an exact copy of the distribution's main repository. Or, more accurately, that's what they are supposed to do.

It turns out, though, that it is easy for anyone, attackers included, to become official mirrors for most major distributions. We tested how easy this was with five of the most popular distributions (Debian, Ubuntu, Fedora, CentOS, and openSUSE). The most that was required for any of them was sending an email announcing the availability of our mirror. In some cases, the process was completely automated and mirror registration was done through a Web site. In the short time we had our mirrors available, we served metadata and packages to thousands of systems, including government and military computers [1].

Once an attacker runs a mirror, it can reply to client requests with malicious content. Most distributions claim to regularly monitor the content of the mirrors to ensure that they are updated and accurate, but this only serves the purpose of ensuring that well-intentioned repositories haven't fallen out of date. It is trivial for malicious mirrors to lie to the monitoring service while still attacking actual clients. (We, of course, did not serve malicious content to clients from our mirrors. The vulnerability testing we did was done privately using our own systems.)

Not every distribution uses public mirrors, though. The commercial Linux distributions SUSE Linux Enterprise and Red Hat Enterprise Linux don't use public mirrors, so these distributions are not at risk from malicious mirrors. They both also use SSL when clients talk to repositories, which protects them from man-in-the-middle (MITM) attacks. (During our research, we discovered a bug in Red Hat Enterprise Linux's use of SSL that still allowed MITM attacks, but Red Hat corrected this very quickly.)

One noncommercial distribution, openSUSE, stands out for using an approach of serving all metadata from its own servers and only allowing

packages to be served from mirrors. This method decreases the risk to users from malicious mirrors, although it doesn't make users completely safe. Why aren't they completely safe? The reason is that even though malicious mirrors are often the easiest way to exploit package managers, there is still the general risk from a MITM attack.

As a MITM, the attacker can control the responses the clients get unless the client talks to the repository using SSL. Very few package managers and distributions support SSL for talking to the repository (currently, only the commercial distributions use SSL). The potential for these MITM attacks comes from a variety of sources, including everything from an attacker on the wire to DNS cache poisoning [4] and BGP prefix hijacking [5].

The fundamental need is to design package managers such that they cannot be exploited by a malicious mirror or a MITM. A package manager needs to be certain that the metadata it receives is accurate and recent, as well as needing to be safe against DoS attacks during the data transfer itself.

Who Was Vulnerable

The more popular package managers and distributions are working on fixing these vulnerabilities. Before looking at the current state of affairs, let's see how these issues impacted the different package managers and distributions. There is some overlap in security mechanisms and vulnerabilities for the various package managers, but we'll try to simplify here.

The first thing to note is that package managers that didn't use signatures on root metadata were all vulnerable, in varying degrees, to metadata manipulation attacks. This included YUM, Portage, and Stork. The "lower down" the signature was placed (closer to the package), the worse the attacks could be. The worst was the case where only packages themselves were signed, as has been the case with YUM. When only packages were signed, all of the metadata manipulation attacks described here were possible.

With replay and freeze attacks, all package managers were vulnerable. However, the package managers that already signed either the root metadata or package metadata required fewer changes to become secure than those that only signed packages. As we'll see in a moment, this has resulted in package managers such as YaST and APT having been more quickly able to focus on protecting against replay and freeze attacks.

The endless data attacks also affected all package managers. These are arguably just implementation flaws, although they are rooted in the more fundamental issue of the client having too much trust in the party it is talking to.

Some smaller distributions have security features available through their package manager but do not use those features. APT-RPM, for example, supports root metadata signatures but PCLinuxOS doesn't use them.

Then there were some other odds and ends. One notable discovery was that Fedora allows users to register mirrors through Fedora's Web site and specify them as responsible for arbitrary CIDR blocks. Attackers can use this to target specific IP address ranges by telling clients in those ranges to communicate with the attacker's mirror. Unfortunately, it appears that Fedora plans to keep this system despite the risks it poses. To understand how serious this is, consider that an attacker could register its mirror as responsible for a block of government or military IP addresses and then attack those computers [6].

Which Package Managers Are Being Secured

The more popular package managers have begun planning and implementing solutions to these vulnerabilities, but currently only a limited amount has been done.

The most vulnerable of the popular package managers, YUM, has added the ability to use signed root metadata, thus securing it from the various metadata manipulation attacks [7]. YUM developers have stated that they plan to warn users if metadata the client receives is too old (that is, a possible replay attack), but this has not yet been implemented. YUM also plans to correct its SSL implementation so that server certificate validity is checked (a similar problem to what RHEL had and fixed) [8].

The ability to sign root metadata will make it into Fedora 10, but it is uncertain whether Fedora 10 will begin making use of this feature. Even more uncertain is when RHEL clone distros such as CentOS might begin using root metadata signatures. The Fedora Project also has stated that it intends to have YUM's initial requests for lists of mirrors be done through SSL. In general, Fedora and YUM developers have expressed their intent to address most issues besides endless data attacks [7].

Debian's APT developers have begun planning the necessary changes to protect APT users from replay attacks. Some of this has made it into Debian's testing branch, but it is not usable yet [9]. Like YUM, APT has also not yet addressed endless data attacks. Ubuntu, being derived from Debian, may follow its lead, although Ubuntu does have its own open bug report for the replay and endless data attacks [10].

With the release of openSUSE 11.1, the openSUSE developers say replay and freeze attacks will be protected against in YaST. They also have said that they are working on protecting against endless data attacks but those changes won't be ready until the following release [11].

Gentoo's Portage developers have begun to address its vulnerabilities. They are in the planning stage for adding a signed root metadata file to Portage, as well as using it to protect against replay and freeze attacks [12]. Additionally, they have already written a patch to protect against endless data attacks.

We implemented protections against all of the attacks mentioned for Stork, our research package management system [13]. We have shared our experiences implementing these mechanisms with other package manager developers to help them implement protections in their package managers.

How to Protect Your Systems

In the long term, the best way you can stay secure against the attacks we've discussed is to choose a distribution that has devoted the necessary time and energy to secure its users against these attacks.

Our findings show that there really is a security advantage to using one of the enterprise distributions, either SUSE Linux Enterprise or Red Hat Enterprise Linux. They fared so well not because they had specifically protected against any of these attacks but, rather, because of their use of SSL for communication and not exposing clients to public mirrors. Among the free distributions, openSUSE should soon offer the same level of protection against these attacks as the enterprise distributions.

In general, the most significant criterion with regard to the vulnerabilities discussed is whether root metadata obtained from the repository is signed. Next, assuming the root metadata is signed, it is important to determine

whether the package manager is able to recognize when the metadata it has obtained is out of date (because of either a stale mirror or a replay attack) and alert you to this without proceeding automatically.

Worth noting is the fact that it's not just a matter of your package manager having this functionality, but also whether your distribution uses it. For example, distributions using YUM do not yet sign root metadata even though the YUM developers acted quickly in adding support for signed metadata. In such cases, you aren't safe until your distribution uses the security options available with the package manager.

In the meantime, if your package manager or distribution is not safe against replay or metadata manipulation attacks, your only option for a high level of security requires additional manual work on your part. You can stay aware of which packages should be updated or installed on your system, invoke your package manager manually, and ensure that the packages you expected to be installed are in fact the ones installed. Organizations running multiple machines would benefit from running their own internal mirror that syncs from a single upstream mirror. All systems on the organization's network can then use the internal mirror for updates. This then requires that only one machine, the internal mirror, be verified to have accurate and updated content (if your sync method or the mirror you sync from is insecure, your network would again be at risk). Keep in mind that if your organization's internal mirror is accessed by machines over a WAN, your machines will still be vulnerable to MITM attacks.

Endless data attacks are important to protect against, as well. This is especially true for mission-critical systems where uptime is of primary importance. If you are using a package manager that is vulnerable to this attack, there are things you can do to protect your systems until your package manager is secured. Most package managers dump the endless data they receive to a file on the file system. Thus, one way to protect the rest of your system against this type of attack is to mount the directory in which the package manager stores downloaded files as its own file system. However, some package managers allow a memory exhaustion DoS through this attack [14]. In those cases, it is probably best to monitor the running of the package manager either manually or by scripting the monitoring of the processes' memory consumption to kill it if necessary. Using an internal mirror accessed only from within your network (as mentioned above) also mitigates this attack.

If your distribution wasn't mentioned or if you are a BSD user, it is important to look into the security of your package management or software update system. If you are a FreeBSD user, for example, we suggest you look into using portsnap. For other distributions, inquire into the security of the update system you are currently using and other more secure options that may exist. The issues we've discussed in this article should provide you with a solid set of security issues to be aware of.

Conclusion and Resources

Securing these systems is a work in progress for all package managers and distributions. We intend to keep a Web site updated with the progress of the various package managers and distributions [15].

If in doubt about the current state of any of them, please take a look at our Web site and don't hesitate to contact us with your questions.

ACKNOWLEDGMENTS

We would especially like to thank Ryan Giobbi at CERT/CC for helping us to responsibly disclose these issues to the package manager maintainers.

Our wholehearted thanks are owed to the many distribution and package manager developers who have discussed these issues with us and who are working on making their systems more secure. In many cases, their openness has helped further our understanding of these issues and the extent of vulnerability of their package managers.

Specifically, we would like to thank Jake Edge, Dag Wieers, Kees Cook, Jan iankko Lieskovsky, Robin Johnson, Ludwig Nussel, Peter Poeml, Marcus Meissner, Josh Bressers, Tomas Hoger, Jason Gunthorpe, Kyricos Pavlou, Mike Piatek, Chris Gniady, Wenjun Hu, and Tadayoshi Kohno for their comments on our research. We also thank the members of the Stork project for their help in creating Stork and their input and work on this research, especially Scott Baker and John Hartman.

REFERENCES

[1] J. Cappos, J. Samuel, S. Baker, and J. Hartman, "A Look in the Mirror: Attacks on Package Managers," *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS '08)*, October 2008.

[2] "Ports Collection": http://en.wikipedia.org/wiki/Ports_collection.

[3] "Secure FreeBSD Ports Tree Updating": http://www.daemonology.net/portsnap/.

[4] US-CERT, "Vulnerability Note VU#800113": http://www.kb.cert.org/vuls/id/800113.

[5] O. Nordström and C. Dovrolis, "Beware of BGP Attacks," *ACM SIGCOMM Computer Communication Review,* 34(2):1–8 (2004).

[6] J. Bressers, "Re: YUM Security Issues...": http://www.redhat.com/archives/fedora-infrastructure-list/2008-July/msg00140.html.

[7] S. Vidal, "[Yum-devel] 3.2.18 Released": http://lists.baseurl.org/pipermail/yum-devel/2008-August/005350.html.

[8] M. Domsch, "Re: YUM Security Issues...": https://www.redhat.com/archives/fedora-infrastructure-list/2008-July/msg00114.html.

[9] "#499897 Preventing Replay Attacks against the Security Archive": http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=499897.

[10] "Bug #247445 in apt (Ubuntu): 'Package managers vulnerable to replay and endless data attacks' ": https://bugs.launchpad.net/ubuntu/+source/apt/+bug/247445.

[11] "Package Management Security on openSUSE": http://lizards.opensuse.org/2008/07/16/package-management-security-on-opensuse/.

[12] http://viewcvs.gentoo.org/viewcvs.py/gentoo/users/robbat2/tree-signing-gleps/.

[13] http://www.cs.arizona.edu/stork/.

[14] J. Cappos, J. Samuel, S. Baker, and J. Hartman, "Package Management Security," University of Arizona Technical Report 08-02, 2008.

[15] http://www.cs.arizona.edu/people/justin/packagemanagersecurity/.