Packages - Bug #3261

[log4j-1.2] Unmaintained, insecure, vulnerable package

2022-04-25 03:48 PM - gap

Status:	confirmed	**% Done:**	0%
Priority:	bug		
Assignee:	GNUtoo		
Category:			

Description

From the upstream URL: https://logging.apache.org/log4j/1.2/

> End of Life
>
> On August 5, 2015 the Logging Services Project Management Committee announced that Log4j 1.x had reached end of life. For complete text of the announcement please see the Apache Blog. Users of Log4j 1 are recommended to upgrade to Apache Log4j 2.
>
> Security Vulnerabilities
>
> Since Log4j 1 is no longer maintained none of the issues listed will be fixed. Users are urged to upgrade to Log4j 2. More issues will be added to this list as they are reported.
>
> CVE-2019-17571 is a high severity issue targeting the SocketServer. Log4j includes a SocketServer that accepts serialized log events and deserializes them without verifying whether the objects are allowed or not. This can provide an attack vector that can be expoited.
>
> CVE-2020-9488 is a moderate severity issue with the SMTPAppender. Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender.
>
> CVE-2021-4104 is a high severity deserialization vulnerability in JMSAppender. JMSAppender uses JNDI in an unprotected manner allowing any application using the JMSAppender to be vulnerable if it is configured to reference an untrusted site or if the site referenced can be accesseed by the attacker. For example, the attacker can cause remote code execution by manipulating the data in the LDAP store.
>
> CVE-2022-23302 is a high severity deserialization vulnerability in JMSSink. JMSSink uses JNDI in an unprotected manner allowing any application using the JMSSink to be vulnerable if it is configured to reference an untrusted site or if the site referenced can be accesseed by the attacker. For example, the attacker can cause remote code execution by manipulating the data in the LDAP store.
>
> CVE-2022-23305 is a high serverity SQL injection flaw in JDBCAppender that allows the data being logged to modify the behavior of the component. By design, the JDBCAppender in Log4j 1.2.x accepts an SQL statement as a configuration parameter where the values to be inserted are converters from PatternLayout. The message converter, %m, is likely to always be included. This allows attackers to manipulate the SQL by entering crafted strings into input fields or headers of an application that are logged allowing unintended SQL queries to be executed.
>
> CVE-2022-23307 is a critical severity against the chainsaw component in Log4j 1.x. This is the same issue corrected in CVE-2020-9493 fixed in Chainsaw 2.1.0 but Chainsaw was included as part of Log4j 1.2.x.

Removing this package would have a knock-on effect on others, which depend on it.

History

#1 - 2022-04-25 03:51 PM - gap

At the least this package should be added to the your-privacy blacklist.

#2 - 2022-04-26 11:04 PM - GNUtoo

Thanks a lot, I've removed it.

Given that there was one critical CVE and 4 high CVEs it's better to do that than keeping it.

#3 - 2022-04-26 11:04 PM - GNUtoo

- Status changed from unconfirmed to confirmed

#4 - 2022-04-26 11:04 PM - GNUtoo

- Assignee set to GNUtoo

#5 - 2022-04-26 11:04 PM - GNUtoo

- Status changed from confirmed to fixed

#6 - 2022-04-26 11:06 PM - GNUtoo

For the details:

- I've removed it from the repositories with db-remove
- I've also removed it from abslibre

I hope that people will still be somewhat noticed of the removal.

#7 - 2022-04-26 11:12 PM - GNUtoo

- Status changed from fixed to confirmed

I'm reopening as bill-auger told me that on IRC:

> < bill-auger> deleting a package from the repos would [not] remove it from anyone's system

#8 - 2022-04-26 11:36 PM - gap

Are there any dependents?
If so, this will have a knock-on effect.

When an installed package is removed from the repos, does pacman tell the user on the next upgrade?
If not, how does the user get notified?

#9 - 2022-04-27 03:07 AM - bill-auger

i noticed pcr/slf4j "A successor to the log4j project"

if that is a suitable replacement, it could be made to 'replaces' the other - from what i understand, the intended use-case for 'replaces', is that pacman will detect if the "replaced" package is installed, and prompt "would you like to replace foo-py with python-foo?" - it is how packages get renamed gracefully

#10 - 2022-05-02 06:22 PM - GNUtoo

As I understand it doesn't affect users already having log4j-1.2 at all (they aren't notified) and their packages continue working. However if they remove, install or reinstall packages depending on log4j-1.2, they won't be able to install such package anymore.

The issue is that there is a lot of dependencies affected.

I've looked rapidly and we have at least:

- java-avalon-framework
- java-avalon-logkit
- jericho-html
- jh
- slf4j
- fop
- diita
- maven
- mysql-jbdc
- antlr2

And in addition jh and maven are probably used in almost all our java packages in libre.

It's still possible to install some of these packages, as sometimes these are build time dependencies though.

The main issue here is that someone needs to fix it and I'm not sure that anyone really has the time to look into it. And having a security issue that big is also an extremely bad idea.

The solution here would be to get more people involved in Parabola and having new people help the current developers fix the huge amount of bugs

that really need fixing (freedom issues) and/or solving the various usability issues.

So the current approach I took for the freedom issues I try to fix is:
(1) To fix the fastest way possible. This typically means removing the package.
(2) To keep the bug open and moving it bug in packaging requests as the bug reports typically have information to make a replacement package
(2) And then I hope that someone will at some point find the time to re-add the fixed package.

Before I was trying to fix the bug well instead of doing it fast (by removing packages) and it didn't really work: I didn't have enough big time chunks to dedicate to fixing bugs, so improvements on 1 package could take several months (my work to fix u-boot did for instance and there is still things to be done). And in the meantime extremely important issues were found and we needed to fix them fast, and existing ones weren't fixed either: many bug reports were kept open for years and nobody did anything on many of the open bugs that were really urgent and problematic.

Once we'll get back to not having no more extremely problematic bugs to fix I'll probably try to get back to trying to bring back packages and so on.

And the good side of that is that if we really try to bring back packages that were removed and so on, at least the problems were gone during all the time gap until that happens.

As for this specific security issue upstream says log4j-1.2 is not maintained anymore, so here I don't think that using log4j-1.2 is the right solution.

So we probably need to rebuild packages to not depend on log4j-1.2 anymore, for instance by using newer versions of log4j that are maintained.

In that case (unless there are bootstrapping issues), keeping log4j-1.2 is probably not really useful to fix the issue.

#11 - 2022-05-02 08:20 PM - gap

Did you read about the recent log4j fiasco where they had a zero-day vulnerability, bearing in mind that's in the **maintained** version?
(https://wikipedia.org/wiki/Log4Shell)

It doesn't seem to the most secure package in the world, and that's the understatement of the decade.

As an OS distribution, we have the job of distributing packages which do not mistreat our users.
I'd argue that it isn't really our job to fix technical problems within packages themselves; that's the job of the maintainers of the package in question.

log4j-1.2 is a security disaster so we should remove it.
By extension, anything which depends on it is also vulnerable and should be removed.
We can't put our users at risk over something this horrifically insecure.
The only thing we can do as a distribution is to remove the offending packages and report the issues upstream.

As a relatively small project we only have the resources to spend time cleaning up packages that are at least maintained and have relatively non-technical issues (eg. freedom, security, privacy, branding, etc.) or issues to do with the packaging of the project, otherwise we're just doing the job of upstream project maintainers for them.

Let's hope for the people who want to use the affected packages that the upstream maintainers can either rebuild against a newer version of log4j, or better yet, switch to something more secure.
Based on its history, I wouldn't install log4j on my system until the project starts doing regular security audits, and I'd recommend it be added to the proposed your-security blacklist if newer versions are packaged.

#12 - 2022-05-02 08:28 PM - gap

As for the issue of pacman not recommending people remove or replace it, I think we should make a security advisory on the RSS news feed.

#13 - 2022-05-02 08:37 PM - GNUtoo

Indeed. I don't know how to do that but the people who already wrote there probably know how to do it. I can write the text to be published if needed though.

#14 - 2022-05-03 12:20 AM - gap

As for the issue of the dependents, can any of them be built without log4j-1.2?
If not, they will need to be removed with the issue that they depend on an insecure package forwarded upstream.

If it can indeed be built without log4j-1.2, I see that the slf4j package is long-outdated and might not even depend on log4j any more, although it probably does.

Most of the packages GNUtoo listed would need to trigger a rebuild on other packages, which could spiral into a humongous dependency graph requiring rebuilds.

I am reminded by this issue that Hyperbola takes issue with Java:
https://wiki.hyperbola.info/doku.php?id=en:philosophy:java_downfalls
and even removed it entirely:
https://www.hyperbola.info/news/end-of-java-openjdk-support/
Following this approach would make our Java-related issues simply melt away.

Moreover, if it were not for the fact that other people use packages that require Java, I would be in favour of this approach.

#15 - 2022-05-03 12:22 AM - GNUtoo

For slf4j I managed to build an updated version without log4j on my laptop, I'll now try to release packages.

What is wrong with Java isn't java, it's mainly maven-central.

#16 - 2022-05-03 12:26 AM - GNUtoo

To be more specific with maven-central, when using a pom.xml, it typically fetches the dependencies from maven-central and not all of them contain all the source code, and many of them probably don't even contain licenses.

At least with older build systems like ant, it was easier but you often had jars bundled in that you needed to remove. And usually the jars came from other projects that also used ant so you could package them more easily.

#17 - 2022-05-03 12:51 AM - GNUtoo

slf4j has been updated to the latest version and doesn't depend on log4j-1.2 anymore.

So the main package pulling all the dependencies has been fixed.

Some still remain as we have the following dependency tree:

```
ditaa -> jericho-html -> log4j-1.2
```

In general a lot of Java packages are also outdated, so they will need to be updated or removed if they are impossible to update (like if the upstream is gone).

#18 - 2022-05-03 03:06 AM - bill-auger

> jh and maven are probably used in almost all our java packages in libre.

i did a bit of digging on the set of packages above, if it helps

java-avalon-logkit and mysql-jbdc do not exist

only fop and ditaa have runtime 'depends'; and those are also in the same set
java-avalon-framework -> fop
jericho-html   -> ditaa

jh is 'makedepends' by 72 packages
slf4j is 'makedepends' by 3 packages
fop is 'makedepends' by 4 packages
ditaa is 'optdepends' by 3 packages
maven is 'makedepends' by 16 packages

> if they remove, install or reinstall packages depending on log4j-1.2, they won't be able to install such package anymore.
> So we probably need to rebuild packages to not depend on log4j-1.2

were there any?

#19 - 2022-05-05 02:34 PM - GNUtoo

bill-auger wrote:

> > jh and maven are probably used in almost all our java packages in libre.
>
> i did a bit of digging on the set of packages above, if it helps
>
> java-avalon-logkit and mysql-jbdc do not exist

That's because I removed java-avalon-logkit and java-avalon-framework very recently, after I wrote that text.

As for myqsl-jbdc I've made a typo, it's mysql-jdbc and there is a package in the feeds at least on i686.

> only fop and ditaa have runtime 'depends'; and those are also in the same set
> java-avalon-framework -> fop
> jericho-html   -> ditaa
>
> jh is 'makedepends' by 72 packages

> slf4j is 'makedepends' by 3 packages
> fop is 'makedepends' by 4 packages
> ditaa is 'optdepends' by 3 packages
> maven is 'makedepends' by 16 packages
>
> > if they remove, install or reinstall packages depending on log4j-1.2, they won't be able to install such package anymore.
> > So we probably need to rebuild packages to not depend on log4j-1.2
>
> were there any?

At the end what worked was to update the packages that depended directly on log4j-1.2 along the way. For slf4j I did it by updating the package. I didn't look if the dependencies were run time or build time because in any case we need to be able to update packages anyway (to fix security issues, and so on).

For jericho-html we probably need to package log4j2. I've started looking at it and it seems relatively similar than the older package for slf4j. However I've a lot of things to do before getting back to fixing that.

For fop, I'm unsure what to do between updating it and removing it: some packages are modified not to depend on it. We could try to update it but it's probably a lot of work. And lot of work also has been done to have it anyway, so it would be sad to loose that work. Currently it's probably broken because I removed the java-avalon-* packages and it depends on them at runtime. I removed it because the upstream stopped maintaining them and points to newer alternatives to them. So projects that depended on avalon also moved on and are now using other dependencies instead.

edit1: Added the fact that I was talking about dependencies in general not run time dependencies + mentioned fop

#20 - 2022-05-07 12:29 PM - bill-auger

> That's because I removed java-avalon-logkit and java-avalon-framework very recently, after I wrote that text.

java-avalon-framework is still in the repos - should it be
deleted?

https://www.parabola.nu/packages/?sort=&q=java-avalon-framework&maintainer=&flagged=

> it's mysql-jdbc and there is a package in the feeds at least on i686.

mysql-jdbc is in the repos for all three arches
https://www.parabola.nu/packages/?sort=&q=mysql-jdbc&maintainer=&flagged=