


## Add script to verify signature #896

 Open

 mz2 opened this issue on 8 Oct 2016 · 13 comments

**mz2** commented on 8 Oct 2016

...

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone





No milestone

Linked pull requests

Successfully merging a pull request may close this issue.


None yet

4 participants



Maybe I am missing something but I cannot find documentation anywhere for how I would verify a signature, given a) a zip / dmg of the update, b) the signature in the form available in an appcast, and the private key that was used to (supposedly) create the signature.


I have a case where I would like to run such a verification on my appcast to find some bad entries where I need to recreate the signature (my app includes an appcast with deltas, and the signatures from the deltas appear to be valid but I appear to have corrupted a few of the signatures for the full-sized updates).

**mz2** commented on 8 Oct 2016

Author

...

I made a quick & dirty command line tool in my Manuscripts.app fork of Sparkle [over here](#) for the purpose.

**zorgiepool** commented on 8 Oct 2016

Contributor


...

Yeah, having such a utility around would be nice. This can also be done using `openssl` in similar spirit to `sign_update` I believe with:

```
/usr/bin/openssl dgst -sha1 -binary < datafile > shafile
/usr/bin/openssl enc -base64 -d < base64sig > sigfile
/usr/bin/openssl dgst -dss1 -verify dsa_pub.pem -signature sigfile shafile
```

1


Like

**mz2** commented on 8 Oct 2016

Author

...

I'm happy to prepare a less hacky version of that command-line tool if given a bit more instructions on the kind of conventions I should follow (command line arguments).

**kornelski** commented on 10 Oct 2016

Member


...

A tool would be nice indeed. I verify my apps in a slightly different way:

1. Edit `.app/Contents/Info.plist` and change version number to an old one
2. Run the app and tell it to update itself

1

Like

**zorgiepool** commented on 10 Oct 2016

Contributor


...

How about a `verify_update` script in `bin/` that takes:

- `update_archive` path to archive to verify
- `dsa_signature` path to file containing base64 encoded signature
- `public_key` path to public key file

We could use our own verification code, but I would like to avoid creating another Xcode target if possible.


[edit]: changed `dsa_signature` to take a path to a file instead

**kornelski** commented on 10 Oct 2016

Member

...

Why would you like to avoid XCode targets?

**mz2** commented on 10 Oct 2016

Author

...

IMO best option would be a build target + a prebuilt copy under the `bin` directory (as it's not production oriented but just something you're going to want to use).


On 9 Oct 2016, at 16:42, Kornel [notifications@github.com](#) wrote:

Why would you like to avoid XCode targets?

—

You are receiving this because you authored the thread.

Reply to this email directly, view it on GitHub, or mute the thread.

**zorgiepool** commented on 10 Oct 2016

Contributor

...

I have found that increasing the number of targets for a project increases management complexity (and in my experience makes merging unpleasant).

Moreover it doesn't have to be objc code or more build configurations we have to maintain if it can be written as a script in the same manner as `sign_update`

In either case I think the signature should be accepted from standard input actually since `sign_update` prints it to standard output.

On Oct 9, 2016, at 2:00 PM, Matias Piipari [notifications@github.com](#) wrote:

IMO best option would be a build target + a prebuilt copy under the `bin` directory (as it's not production oriented but just something you're going to want to use).

On 9 Oct 2016, at 16:42, Kornel [notifications@github.com](#) wrote:

Why would you like to avoid XCode targets?

—


You are receiving this because you authored the thread.

Reply to this email directly, view it on GitHub, or mute the thread.

—

You are receiving this because you commented.

Reply to this email directly, view it on GitHub, or mute the thread.

**mz2** commented on 10 Oct 2016

Author

...

Sure, but a) this target would have to be touched almost never and b) if changes were to be made to the signing code, then nothing needs updating, and related to b, c) the validation is exercising exactly the same validation code as what Sparkle uses in production.

On 9 Oct 2016, at 20:32, Mayur Pawashe [notifications@github.com](#) wrote:

I have found that increasing the number of targets for a project increases management complexity (and in my experience makes merging unpleasant).

Moreover it doesn't have to be objc code or more build configurations we have to maintain if it can be written as a script in the same manner as `sign_update`

In either case I think the signature should be accepted from standard input actually since `sign_update` prints it to standard output.

On Oct 9, 2016, at 2:00 PM, Matias Piipari [notifications@github.com](#) wrote:

IMO best option would be a build target + a prebuilt copy under the `bin` directory (as it's not production oriented but just something you're going to want to use).

On 9 Oct 2016, at 16:42, Kornel [notifications@github.com](#) wrote:

Why would you like to avoid XCode targets?

—

You are receiving this because you authored the thread.

Reply to this email directly, view it on GitHub, or mute the thread.

—

You are receiving this because you commented.

Reply to this email directly, view it on GitHub, or mute the thread.

—

You are receiving this because you authored the thread.

Reply to this email directly, view it on GitHub, or mute the thread.

**zorgiepool** commented on 10 Oct 2016

Contributor

...

Good points, but I think the code may not be purposeful enough to outweigh the cost of adding an additional build target (I think a cost is incurred by just having it around); I think the objc code would be more likely to need updating than a shell script as well.

IMO, having `sign_update` and `verify_update` be consistently written may be more important than code sharing. There would now also be different system requirements between the two if the verification was a compiled program, which can be easily avoided.

I think it may be very useful to know if verification output via `openssl` differs from our Sparkle verification code actually. It may worrisome enough to add additional unit testing.

**mz2** commented on 10 Oct 2016

Author

...

Good points, but I think the code may not be purposeful enough to outweigh the cost of adding an additional build target (I think a cost is incurred by just having it around); I think the objc code would be more likely to need updating than a shell script as well.

Feels counter-intuitive: if you look at the code I wrote, it makes exactly one call to a public Sparkle API call, and I did it this way so I can be sure what I am validating is exactly the same thing as what Sparkle does, and because it's the simplest possible implementation I could think of (I tried and failed to bash something together with openssl, resulting in various cryptic error messages, before realising this as a simple way to go). If anything, would be nice to move away from OpenSSL for `sign_update` rather than add new dependencies to it.

Setting OpenSSL command line interface and its history and deprecation status in Apple's SDK aside, the easiest way to maintain such a tool in my mind is for a build system to tell if something's changed in a way that it would no longer build, rather than to sometime in the future notice that bitrot has happened? Hacking something together in bash that does not visibly break on platform changes does not mean it couldn't somehow break (breakage just wasn't observed). If a tool like this is considered handy enough to be included in the repo, what would be most useful IMO would be to review the build settings for that two source code file target are sensible and that the command line interface and error handling suffice for the purposes of a diagnostic tool, rather than writing the same thing in an interpreted shell scripting language that depends on openssl beside what Apple's SDK provides.

IMO, having `sign_update` and `verify_update` be consistently written may be more important than code sharing. There would now also be different system requirements between the two if the verification was a compiled program, which can be easily avoided.

The system requirements for the tool are those of Sparkle.framework (well, less requirements really, it just compiles in DSAVerifier, as it is itself dependency free I figured this would be cleaner than linking the framework). There are no new sets of system requirements as far as I can tell, just project level configuration should be followed for that target. I treat it as a feature that I don't need OpenSSL to be available for this to work, not as an additional set of requirements (fully realising `sign_update` is openssl dependent atm, doesn't mean its counterpart should have that additional dependency).

I think it may be very useful to know if verification output via openssl differs from our Sparkle verification code actually. It may worrisome enough to add additional unit testing.

It's not that I observed a difference, and the expectation on seeing a difference surely would be that the build is corrupt or the validation tool is broken – I was simply unable to put together a script with OpenSSL after some google assisted searching and found writing a single DSAVerifier API call utilising objc program therefore infinitely easier to write. My goal was to get a good confidence that it checks the same thing as Sparkle in production with a simple tool, I believe what I wrote meets that need well and does not really in any concrete terms cause a maintenance burden.

—

You are receiving this because you authored the thread.

Reply to this email directly, view it on GitHub, or mute the thread.

**zorgiepool** commented on 10 Oct 2016

Contributor

...

My point that it is not very much code also simulated by a built-in tool is why I got the feeling that it may have not been worthwhile enough to create a whole target for. I realize that figuring out how to do it with the command line tool may have been challenging, which is partly why I pasted commands showing how to achieve it. That alone does not mean a new target is the way to go.

The other day someone here was running `sign_update` on a linux system, which was why I felt the need to bring up system requirements, but I do wonder if that's worthwhile caring about.

I think (note: as a final product) that a script could currently be the simplest change to accept into the overall project. However I do realize the same verification code is not being exercised, and maybe using openssl isn't ideal, but it's kind of unfortunate to have our tools result to different methods.

**kornelski** mentioned this issue on 29 Dec 2016

generate\_appcast should verify that updates pass signature checks #955

 Open

**pointum** commented on 27 Oct 2017

...

I've stumbled on this issue so here's a `sign_verify` script with no temporary files based on [@zorgiepool suggestion](#) and `sign_update` in case someone else needs this:

```
#!/bin/bash
set -e
set -o pipefail
if [ "$#" -ne 3 ]; then
  echo "Usage: $0 public_key_path update_archive_path base64_signature"
  exit 1
fi
openssl=/usr/bin/openssl

$openssl dgst -sha1 -binary < "$2" | $openssl dgst -sha1 -verify "$1" -signature <(echo "$3" | $openssl enc -base64 -d)
```

2

Like

**dmoagx** mentioned this issue on 3 Apr 2018

Verify download sequelpro/sequelpro#3024

 Closed

 Sign up for free

 to join this conversation on GitHub. Already have an account? [Sign in to comment](#)