

# Secure by default sessions #2200

New issue

🔗 Open

stigtsp wants to merge 1 commit into [mojolicious:main](#) from [stigtsp:secure-by-default-sessions](#) 📄

💬 Conversation 76    🔗 Commits 1    📄 Checks 11    📄 Files changed 17    +278 -26 🟢🟢🟢🟢🟢






















stigtsp commented on Sep 28, 2024 • edited ▼

## Summary

To make session secrets secure by default, generate and persist a 256 bit session secret:

- Add `urandom_bytes` and `urandom_urlsafe` to `Mojo::Util` for generating secure random bits from either `Crypt::Random` OR `/dev/urandom`
- Don't use the hard coded moniker as the default secret
- Generate and store a strong secret if not exists in `$ENV{MOJO_HOME}/mojo.secrets` , overridable with `$ENV{MOJO_SECRETS_FILE}` when `app->secrets` is called
- Only load secrets from `mojo.secrets` that are over 22 chars
- Use `urandom_urlsafe` when generating CSRF tokens
- Use `urandom_urlsafe` in `mojo generate app`
- Add `mojo generate secret`
- Tests:
  - Add misc tests for generating and loading `mojo.secrets` in `t/mojolicious/secret/` and for `mojo generate secret` .
  - Add a default secret in `t/mojolicious/mojo.secrets` so other session checks work
- Install `Crypt::URandom` in GH Windows workflow so `urandom_bytes` works on that platform

## Reviewers

-  **krai** 
-  **robrwo** 
-  **latk** 
-  **jkramarz** 
-  **guest20** 
-  **marcusramberg**  
-  **christopherraa**  
-  **Grinnz**  

At least 2 approving reviews are required to merge this pull request.

## Assignees

No one assigned

## Labels

None yet

## Projects

None yet

## Milestone

No milestone

## Development

Successfully merging this pull request may close these issues.

## Also consider

- Disallowing insecure secrets from being passed to `app->secrets()`
- Removing padding of HMAC message to 1025 bytes introduced by [🔗 pad session values to 1025 bytes by default #1791](#)

## Motivation

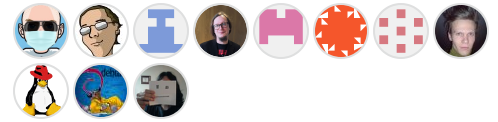
Making HMAC protected sessions secure by default after discussing with [@jberger](#) at PTS24.

It has been demonstrated by [Baking Mojolicious cookies](#) and the recent discussion in [#1791](#) that this is not the case.



None yet

11 participants



**stigtsp** mentioned this pull request on Sep 28, 2024

**pad session values to 1025 bytes by default #1791**

Merged



**Grinnz** reviewed on Sep 28, 2024 [View reviewed changes](#)

lib/Mojo/Util.pm

Outdated

Show resolved

lib/Mojolicious.pm

```
64 | +
65 | + $self->log->trace("Your secret passpl
66 | +
67 | + return [$secret];
```



**Grinnz** on Sep 28, 2024

Contributor

While I think this is generally a good and necessary step, this will invalidate session cookies in currently insecure apps so that will need to at minimum be noted in the changelog (which is not the responsibility of this PR).

lib/Mojolicious.pm

Outdated

Show resolved

lib/Mojolicious/Command/

Author/generate/secret.p

Outdated

Show resolved

m

t/mojo/util.t Outdated

Show resolved



guest20 reviewed on Sep 28, 2024 View reviewed changes

lib/Mojo/Util.pm Outdated

Show resolved



Grinnz reviewed on Sep 28, 2024 View reviewed changes

lib/Mojo/Util.pm Outdated

Show resolved

kraih requested review from a team, marcusramberg, kraih, christopherraa and Grinnz 8 months ago



kraih commented on Sep 28, 2024

Member

Make sure to squash your comits when you are ready.



1



latk reviewed on Sep 28, 2024 View reviewed changes

lib/Mojo/Util.pm Outdated

Show resolved



stigtsp commented on Sep 28, 2024

Author

Thanks! I'm addressing all feedback, and will update the PR tomorrow or so.



1



kraih reviewed on Sep 28, 2024 View reviewed changes

lib/Mojo/Util.pm Outdated

```
95 | + require Win32::API;
96 | +
97 | + my $RtlGenRandom = Win32::API->new('a
98 | + or croak "SystemFunction036 import
```

**kraih** on Sep 28, 2024

Member

Is this actually used in our Windows CI tests? This seems like code nobody on the team can maintain long term and we will end up removing it because of that the first it causes problems.

**stigosp** on Sep 29, 2024

Author

Tests are running `SystemFunction036` (`RtlGenRandom`) on Windows to get random bytes. It's used in the same way as for example [Crypt::URandom](#). However, It is a legacy API, so we can consider using `ProcessPrng` instead.

Here are some discussions in rust and go lang:

- [✔ Call RtlGenRandom\(\) instead of CryptGetRandom\(\) on Windows](#) [rust-random/rand#111](#)
- [✔ crypto/rand: Legacy RtlGenRandom use on Windows](#) [golang/go#53192](#)

**kraih** on Sep 30, 2024

Member

This part of the code will probably end up being a blocker, since i don't see anyone on the team agreeing to take responsibility for maintaining it in the future.

**jkramarz** on Sep 30, 2024

I'd not expect finding such code in web framework. In the same time, AFAIK the alternatives are:

- using some small old library with single maintainer and with no recent releases, that does exactly this call, but conveniently wrapped,
- overkill of making `Net-SSLLeay` a dependency to get it through `openssl` binding.

**kraih** on Sep 30, 2024 • edited ▼

Member

`Net::SSLLeay` is already a soft dependency, maybe there's a clean way to make this feature optional? A hard dependency is pretty much out of the question. It's unfortunate `Perl` doesn't ship with some kind of `OpenSSL` binding, for `mojo.js` we've chosen a different solution and [just encrypt all sessions](#) because `Node` has `crypto` support.



**guest20** on Sep 30, 2024 • edited ▼

`Crypt::URandom` has a pile of different window support calls. Adding that as an optional dependency for windows enjoyers might unblock this, while outsourcing the windows support



**stigts** on Sep 30, 2024 • edited ▼

Author

`Crypt::URandom` appears to [sets up some context](#) for `CryptGenRandom` for Win2k, and uses `RtlGenRandom` for later Windows versions.

According to a discussion in golang crypto/rand, it seems fine to use `RtlGenRandom` (SystemFunction036) [golang/go#33542 \(comment\)](#)


We could use `Crypt::URandom` as a soft dependency for Win and Linux/UNIX, and fail over to `/dev/urandom` if that module is not installed? (And fatal on Windows maybe)



**stigts** on Oct 1, 2024 • edited ▼

Author

What about something like this?

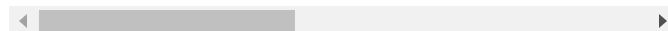
```
use constant CRYPT_URANDOM => !!eval {  ir
...

sub urandom_bytes {
    my $num = shift || 32;

    return Crypt::URandom::urandom($num) if CR\
croak 'Cannot find /dev/urandom. On Windows
    unless -f '/dev/urandom';

    open(my $urandom, '<', '/dev/urandom') or
    sysread($urandom, my $bytes, $num) == $num
    close($urandom);

    return $bytes;
}
```



**guest20** on Oct 1, 2024

That's exactly the kind of shim(affectionate) I was suggesting!

You might not need to mention windows directly in the message, you know, just to be inclusive of that one person who runs mojo apps on VAX or AIX



**stigtsp** on Oct 2, 2024

Author

I've updated the PR with logic similar to the above, avoiding any Win32 specific code.

Also added `crypt::URandom` to the Windows GH actions workflow, should the be in Makefile.PL instead?



**kraiH** reviewed on Sep 28, 2024

[View reviewed changes](#)

lib/Mojo/Util.pm

Outdated

[Show resolved](#)



**kraiH** reviewed on Sep 28, 2024

[View reviewed changes](#)

lib/Mojolicious/Command/Author/generate/secret.pm

[Show resolved](#)



**kraiH** reviewed on Sep 28, 2024

[View reviewed changes](#)

lib/Mojolicious.pm

Outdated

[Show resolved](#)



**kraiH** reviewed on Sep 28, 2024

[View reviewed changes](#)

lib/Mojolicious.pm

Outdated

[Show resolved](#)



**kraiH** reviewed on Sep 28, 2024

[View reviewed changes](#)

t/mojolicious/secret/lite-create.t

Outdated

[Show resolved](#)



**stigtsp** [force-pushed](#) the `secure-by-default-sessions` branch 3 times, most recently from `c33c0eb` to `6bd013a` 8 months ago

[Compare](#)



**stigtsp** commented on Sep 29, 2024

Author

I've (hopefully) addressed all comments and pushed a single commit to this PR.

One problem that remains is that a `mojo.secrets` file is left over in the `t/mojo/` directory after tests are run, likely due to something calling `app->secrets` there. A quick fix for this is to copy the one from `t/mojolicious/` but it seems a bit messy. Any better ideas on how to fix this?



**latk** reviewed on Sep 29, 2024

[View reviewed changes](#)

lib/Mojo/Util.pm

Outdated

Show resolved

lib/Mojo/Util.pm

Outdated

Show resolved

lib/Mojolicious/Command/  
Author/generate/secret.p  
m

Outdated

Show resolved

lib/Mojolicious.pm

Comment on lines +52 to +54

```
52 + # Read secrets and filter out those  
53 + # (~128 bits), as they are not like  
54 + my @secrets = grep { length $_ >= 128 }
```



**latk** on Sep 29, 2024

Should there be a warning message if a candidate secret is skipped? Right now, they're discarded silently.



**stigtsp** on Sep 30, 2024

Author

Yes - or maybe even a fatal error?

Btw, I'm not sure how useful more than one key in `app->secrets` is. With a strong secret, wouldn't rotation only makes sense if it is compromised/leaked/etc? And then you wouldn't want to continue trusting the old key imho.



**Grinnz** on Sep 30, 2024

Contributor

It is useful so that instances using the session key can be upgraded to use the new secret over even a short period of time without immediately invalidating everything.



**Grinnz** on Sep 30, 2024

Contributor

Also, regular secret rotation is good practice even when not found to be compromised.



**guest20** on Oct 1, 2024

I think only the first secret (the one used to set new session cookies) should be subject to the length restriction, since you might still want to restore sessions cookies that were set with crappy secrets so you can re-issue them with new stronger key...



**stigtsp** force-pushed the `secure-by-default-sessions` branch 4 times, most recently from `0cbcf0d` to `53b7107` 8 months ago

Compare



Generate and persist a 256 bit session secret by default ...

Verified

✓ fb4ae3f



**stigtsp** force-pushed the `secure-by-default-sessions` branch from `53b7107` to `fb4ae3f` 8 months ago

Compare



**stigtsp** marked this pull request as ready for review 7 months ago



**kraith** commented on Oct 11, 2024

Member

Heads up, early November is SUSE hack week, and i might work on a port of the `mojo.js` encrypted session implementation to Mojolicious. That's probably the best solution for security.



**daleif** commented on Oct 11, 2024 via email

[like] Lars Madsen reacted to your message:

...





**jkramarz** commented on Oct 11, 2024

Heads up, early November is SUSE hack week, and i might work on a port of the mojo.js encrypted session implementation to Mojolicious. That's probably the best solution for security.

Assuming that we're talking about <https://github.com/mojolicious/mojo.js/blob/436b3263690efe65b0a8b4ecbbb09e4a3c99af5e/src/session.ts#L77> , please consider if it's not vulnerable to IV nonce reuse ([https://frereit.de/aes\\_gcm/](https://frereit.de/aes_gcm/)) and is using scrypt key derivation algorithm correctly (<https://nodejs.org/api/crypto.html#cryptoscriptpassword-salt-keylen-options-callback> , see notes about choosing proper salt).

If considering change of session cookies format, what about just using Crypt::JWT library instead?



**kraiH** commented on Oct 11, 2024

Member

If considering change of session cookies format, what about just using Crypt::JWT library instead?

Good idea. Worth investigating.



**guest20** commented on Oct 11, 2024

I'm not sure if JWT gives us much more than an extra header section with:

- "which crypto to use" field (a source of vulns of honoured) and
- "type" which is ... "jwt"



**latk** commented on Oct 11, 2024

i might work on a port of the mojo.js encrypted session implementation to Mojolicious. That's probably the best solution for security.

Even if the session cookie format is changed, a secure method for generating secrets is needed, as provided by this PR.

Having strong keys so that the cryptographic methods can uphold their security guarantees may be more important than deciding whether to offer only Integrity (via the current use of a HMAC), or Integrity+Confidentiality (by switching to an authenticated encryption method like AES-GCM).

If porting the JavaScript code, it could be made significantly more efficient:

```
static async encrypt(secret: string, value: string) {
  const key = await scrypt(secret, 'salt', 32);
  const iv = await randomBytes(12);
  const cipher = crypto.createCipheriv('aes-256-gcm',
```

1. I am not convinced that this use of `scrypt()` has any security benefits over a fast cryptographic hash function like SHA-2, especially if the `secret` is already strong.
2. Due to using a constant salt, the `scrypt()` call could be pre-computed during application startup, instead of re-running this for each `encrypt()` and `decrypt()`.

Mojo.js doesn't have benchmarks involving session cookies, but if it had some then caching (or removing) the `scrypt()` call might save on the order of 100ms per request.

what about just using `Crypt::JWT` library instead?

+1 on sticking close to standard formats like JWT, as this neatly sidesteps many pitfalls from doing low-level crypto manually. (Aside from the weakness that a JWT decoder might fail to limit which algorithms are accepted.)

That particular library has a hard dependency on `cryptX`, which provides a `CryptX::PRNG::random_bytes()` function which would remove the need to manually open `/dev/urandom`. This would also solve the problem of a secure fallback on Windows.

But pulling in a large and security-critical XS dependency just for that might not be worth it? The current (and much simpler) approach to session cookies can already be secure (if strong keys are used).





**kraih** commented on Oct 12, 2024

Member

Even if the session cookie format is changed, a secure method for generating secrets is needed, as provided by this PR.

Is it really? Why can't we outsource that problem to plugins on CPAN? Any core mojo APIs missing?



**stigtsp** commented on Oct 12, 2024

Author

Even if the session cookie format is changed, a secure method for generating secrets is needed, as provided by this PR.

Is it really? Why can't we outsource that problem to plugins on CPAN? Any core mojo APIs missing?

**@kraih** Yes. Strong keys are required for HMAC signed cookies. I have no strong opinion on what CPAN modules or syscalls are used for randomness, as long as bits are from a csprng.

For example, NIST requires HMAC keys to be a minimum of 128 bits from a cryptographic random number generator:  
<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-224.ipd.pdf>

Problems:

1. A well known string is used as default HMAC key. Even though its documented, and a warning is printed, it provides no security and could be a vulnerability if it ends up in production. [CWE-321](#)
2. No key length requirement for users when setting keys. This leads to weak keys in the wild vulnerable to dictionary or brute force attacks, as described in [Baking Mojolicious cookies](#) by Antoine Cervoise. [CWE-521](#)
3. `mojo generate app` generates a weak secret using `sha1_sum $$ . steady_time . rand`, which is not from a cryptographic number generator. [CWE-338](#)

This PR aims to start fixing these problems, so let me know if there is anything more I can do to help.



2



**kraih** commented on Oct 12, 2024

Member

**@kraih** Yes. Strong keys are required for HMAC signed cookies. I have no strong opinion on what CPAN modules or syscalls are used for randomness, as long as bits are from a csprng.

Perhaps there was a miscommunication then. We make no claim to be secure by default, that's why we ask users to change their secrets and suggest to rotate regularly. The secrets we generate by default are strictly for development purposes, just like the TLS certificate we ship. It is the users responsibility to change both for production.

The default secrets being better with an optional CryptX dependency because of an alternative session implementation would merely be a positive side effect, not an end goal.



**jkramarz** commented on Oct 12, 2024

Out of curiosity, I checked how some top starred projects on Github with direct Mojolicious dependency are dealing with this problem:

- zonemaster and mojolicious.org website has no sessions at all :-)
- Convos is using sha1 of (usually) some random bytes from /dev/urandom: [convos-chat/convos@ 54d1763 #diff-14d3feb01f623517c2b3475efcc7cdbcc32705fa461ab5c2ecda1d529f627ff4R134](https://github.com/convos-chat/convos/commit/54d1763#diff-14d3feb01f623517c2b3475efcc7cdbcc32705fa461ab5c2ecda1d529f627ff4R134)
- LANraragi is using bytes from Perl non-cryptographic rand, then storing it in (probably) web-accessible temp directory: [Difegue/LANraragi@ 7ba8425](https://github.com/Difegue/LANraragi/commit/7ba8425)
- Yancy hopes there's secret defined: <https://github.com/preaction/Yancy/blob/83ae73dc31aa0f72cf5c39b79df8a121c9370385/lib/Yancy/Plugin/Auth/ToKen.pm#L239>
- openSUSE's MirrorCache is using a string formed with Perl non-cryptographic rand and not storing it at all: <https://github.com/openSUSE/MirrorCache/blob/c8d8d53b3ec522d8372f44c807af11fa47fbee7a/lib/MirrorCache/WebAPI.pm#L88>
- Ado hopes that adding some magic sprinkles works: <https://github.com/kberov/Ado/blob/a93056a50a8719b4c5204444c3b929e3b4716516/lib/Ado.pm#L51>

- Mojowka has a static string:  
<https://github.com/shoorick/mojowka/blob/0159a8e8e5810836c91a4d6d5be8e00e9a634373/mojowka#L244>
- USGCRP's gcis won't start without proper configuration (and I hope it's not `this_should_be_replaced_with_a_server_secret` in production deployment, but didn't dare to check):  
<https://github.com/USGCRP/gcis/blob/b291522f1ab7cfff850cb5f25a7491039d04f6c5/lib/Tuba.pm#L46>



**s1037989** commented on Oct 25, 2024

Contributor

This seems like a really neat solution and appears very Mojo. The PR looks really well made and followed all the contributing guidelines. I'd love to see this merged. ... are there any cons to merging this in?



**kraih** commented on Oct 25, 2024

Member

I'm more and more leaning against this proposal. Especially the secret files don't feel very Mojolicious. We'll see after encrypted sessions have been implemented.



**christopherraa** commented on Oct 25, 2024

Member

Personally I am 🙅 on file based solutions like the one proposed. Any writing to disk also introduce a potential additional attack-vector and on top of that deviates from the move we've been seeing in recent years towards more and more env-based approach to configuration of services, especially for containerized workloads. When doing application hardening and insulation we actively try to limit the applications access to and use of the file system, except where strictly necessary.

Those were my \$.02



**guest20** commented on Oct 26, 2024

**@christopherraa** If the secret isn't persisted, all cookies are invalidated every time the application restarts. The only other choice is making the secret mandatory and aborting startup if it's not set... which gets awful CrashLoopBackoff'y



**Grinnz** commented on Oct 26, 2024

Contributor

Not to mention that there are numerous applications out there which have never touched the session cookie or at least have not used it for anything sensitive. I personally think it is very important that it be secure by default, but I also have reservations about writing to disk for permissions/deployment reasons; but it would go along with the default behavior of hypnotoad to write a pidfile to the project directory if not configured otherwise. (The default behavior of the application log to log to the project directory has been removed, as it simply logs to STDERR by default now)



**krai** commented on Oct 26, 2024

Member

I have to agree with [@christopherraa](#) that ephemeral parts of the config, such as secrets, need to be easily updatable in container environments.



**s1037989** commented on Oct 26, 2024

Contributor

What about using a machine-generated unique ID, and then fall back on the moniker if the file can be read?

`/var/lib/dbus/machine-id`



**krai** mentioned this pull request on Nov 21, 2024

**Add support for encrypted sessions with CryptX #2212**

Merged



**krai** commented on Nov 21, 2024

Member

[#2212](#) is what i had in mind for encrypted sessions.



**robrwo** reviewed on Nov 21, 2024

[View reviewed changes](#)

```
lib/Mojo/Util.pm
```

```
386 | +  
387 | + return Crypt::URandom::urandom($num)
```

388 +  
389 + croak 'Cannot find /dev/urandom, ins



**robrwo** on Nov 21, 2024

Contributor

Is it worth having a fallback to read /dev/urandom  
vs simply requiring the user to have this module  
installed if they are using this feature?



**mergify** bot commented on Nov 22, 2024

Contributor

This pull request is now in conflicts. Could you fix it [@stigtsp](#)?



**carnil** commented 5 hours ago

CVE-2024-58135 is related to this merge request.

<https://lists.security.metacpan.org/cve-announce/msg/29241187/>