PHP Becomes First Programming Language To Add 'Modern' Cryptography Library In Its Core

(bleepingcomputer.com)

Posted by BeauHD on Tuesday February 21, 2017 @05:00PM from the center-of-the-earth dept.

An anonymous reader writes from a report via BleepingComputer: *The PHP team has unanimously voted to integrate the Libsodium library in the PHP core, and by doing so, becoming the first programming language to support a modern cryptography library by default. Developers approved a proposal with a vote of 37 to 0 and decided that Libsodium will be added to the upcoming PHP 7.2 release that will be launched towards the end of 2017. Scott Arciszewski, the cryptography expert who made the proposal, says that by supporting modern crypto in the PHP core, the PHP team will force the WordPress team to implement better security in its CMS, something they avoided until now. Additionally, it will allow PHP and CMS developers to add advanced cryptography features to their apps that run on shared hosting providers, where until now they weren't able to install custom PHP extensions to support modern cryptography. Other reasons on why he made the proposal are detailed here. Arciszewski also says that PHP is actually "the first" programming language to support a "modern" cryptography library in its core, despite Erlang and Go including similar libraries, which he claims are not as powerful and up-to-date as PHP's upcoming Libsodium implementation.*

library php software

→

Health Apps Could Be Doing More Harm Than Good, Warn Scientists
Ask Slashdot: What Are Some Bad Programming Ideas That Work?
Developer Argues For 'Forgotten Code Constructs' Like GOTO and Eval
Are Flawed Languages Creating Bad Software?
Linus Torvalds In Sweary Rant About Punctuation In Kernel Comments
Will The Death of the PC Bring 'An End To Openness'?
Submission: PHP Is First Language To Add "Modern" Cryptography Library To Its Core
UPS Develops 'Rolling Warehouse' System In Which Drones Are Launched From Atop Trucks

PHP Becomes First Programming Language To Add 'Modern' Cryptography Library In Its Core 75 More | Reply Login

PHP Becomes First Programming Language To Add 'Modern' Cryptography Library In Its Core

Post | Load All Comments

Search 75 Comments | Log In/Create an Account

Comments Filter:
- All
- Insightful
- Informative
- Interesting
- Funny

The Fine Print: The following comments are owned by whoever posted them. We are not responsible for them in any way.

Oh please (Score:5, Insightful)
by Anonymous Coward on Tuesday February 21, 2017 @05:07PM (#53907929)
Any language where the default equality comparison operator is *true* given two string-type variables with values "0E54321" and "0E12345" is not a cryptographically secure language. In fact there is a nonzero chance of the default equality operator returning true between two different MD5 or SHA256 hashes if they happen to fall into a hexadecimal form that is all digits except for one E or F.
Anyone who ... somehow more secure as a language because it has added *new optional library calls* without doing anything about the fundamental language defects is demented.

Flag as Inappropriate
2 hidden comments

- **Re: Queue the Rust is better people (Score:2)**
by cyber-vandal (148830)
Cue*

- **Re: (Score:2)**
by PCM2 (4486)
Well, technically you could have them line up for comment.

- **Re:Oh please (Score:5, Informative)**
by Vairon (17314) on Tuesday February 21, 2017 @05:43PM (#53908121)
PHP has a comparison operator === that evaluates if the two things it is comparing are equal and of the same type.
$ php -r "if (\"0E54321\" === \"0E12345\") { echo 'equal'; } else { echo 'not equal'; } "
not equal
Without ===, variable type conversion can cause a string containing numbers to be converted to an integer. See these links for details:
http://php.net/manual/en/langu... [php.net]
http://php.net/manual/en/langu... [php.net]
Reply to This | Parent | Share
twitter facebook linkedin
Flag as Inappropriate

- **Re:Oh please (Score:4, Interesting)**
by Lisandro (799651) on Tuesday February 21, 2017 @05:54PM (#53908185)
Except for objects, where a === b is true only if a and b *are the same object*. Such a beautifully designed language.
Reply to This | Parent | Share
twitter facebook linkedin
Flag as Inappropriate

- **Re: (Score:2)**
by deKernel (65640)
Please tell me you are kidding me on this? I haven't touched PHP so this is all new to me...if it is true.

- **Re: (Score:3)**
by Lisandro (799651)
Oh, how i wish i was: https://developers.slashdot.or... [slashdot.org]

- **Re: (Score:2)**
by TheRaven64 (641858)
That's pretty common for OO languages (or, in fact, any language with a notion of subtyping), where individual classes implement their own comparison operators. In C++ you can overload the comparison operators, but most OO languages that don't do operator overloading just use a named method. If you write a.equals(b), then you'll call the equals method implemented by the class of a. If you write b.equals(a) then you'll call the method implemented by the class of b. One may know about the other, but the c

- **Re: (Score:1)**
by Anonymous Coward
Funny how the rest of the languages out there which do type conversion manage to avoid this shit: http://phpsadness.com/static/p... [phpsadness.com]

- **Re: (Score:2)**
by hcs_$reboot (1536101)
Except for objects, where a === b is true only if a and b *are the same object*
In Java, a == b is true if a and b references are identical, which is probably what does PHP under the hood.

- **Re: Oh please (Score:1)**
by p91paul (4513273)
Object comparison is not easy, you have to compare every field and if fields are objects, compare their fields and so on. And I did not take into account recursive objects, where an object contains a reference to itself, either directly or through a contained object. Nor thread safety: what should happen if an object is modified while the comparison (which is at this point not atomic) occurs? Hiding all that in a == would be very bad. And well, objects are pointers, so with == you are comparing them (if you

- **Re: (Score:2)**
by Lisandro (799651)
Oh, it certainly isn't. Too bad on PHP, == when used on objects compares both value and type of every attribute in them, which is what === does for all other types.

- **Re: (Score:2)**
by gweihir (88907)
Looks like they were trying to come up with a competitor for "brainfuck" and then too many people did not get the joke and though this was a language intended fro real work...

- **Re: (Score:1)**
by Anonymous Coward
B===D is much better than B=D. Trust me, I'm an engineer.

- **Re: (Score:2)**
by hcs_$reboot (1536101)
even the double equals of C is one of the *least* "beautiful" design decisions ever made.
It's not because some not so good programmers confuse '=' with '==' that '==' is not "beautiful". On the contrary, I think this is one of the many genius ideas K&R had at the time ('a = b', 'a == b'... think about it!)

- **Same type inequality comparison operators? (Score:2)**
by tepples (727027)
Is there also a "less than and of the same type" operator? Or is calling `strcmp()` the best practice for this?

- **Re: (Score:1)**
by hcs_$reboot (1536101)
`if ("0E54321" === "0E12345") { echo 'equal'; } else { echo 'not equal'; }`
Too bad your example is useless (like 80% of PHP's doc examples anyway), since `if ("0E54321" == "0E12345")` (only 2 '=') yields also 'not equal'. A more relevant example is `if ("123" === 123)` (yields false but '==' yields true).

- **Re: Oh please (Score:5, Informative)**
by Edgewize (262271) on Tuesday February 21, 2017 @07:45PM (#53908629)
Or more relevantly, I think this is what the original poster was referring to:
https://www.whitehatsec.com/bl... [whitehatsec.com]
Here are some examples of PHP doing mind boggling things with md5 and sha1 hashes.
https://3v4l.org/tT4l8 [3v4l.org]
Reply to This | Parent | Share
twitter facebook linkedin
Flag as Inappropriate

- **Re: (Score:2)**
by JustAnotherOldGuy (4145623)
Without ===, variable type conversion can cause a string containing numbers to be converted to an integer.
Yep, and as much as I like php, I'll be the first to say that php's "converting a string to an integer" as a default behavior is idiotic. It's just an accident waiting to happen.

- **Re: (Score:3)**
by gweihir (88907)
It is a sign of very junior and inexperienced language designers with very big egos. And no concept of personal responsibility.
Writing good code is hard enough with good tools.

- **Re:Oh please (Score:4, Funny)**
by fahrbot-bot (874524) on Tuesday February 21, 2017 @07:43PM (#53908625)
PHP has a comparison operator === that evaluates if the two things it is comparing are equal and of the same type.
The next version will support "====" for things are really, *really* equal.
Reply to This | Parent | Share
twitter facebook linkedin
Flag as Inappropriate

- **Re: (Score:2)**
by sconeu (64226)
It was needed for Animal Farm Simulation support, because some objects are more equal than others.

- **Re: (Score:1)**
by hcs_$reboot (1536101)
Yes. Now go ahead and guesstimate the percentage of major PHP software packages accidentally using == where the developer should have used ===. Whose fault is that?

- **Re: (Score:2)**
by TheRaven64 (641858)
If one developer makes a mistake, it's the fault of the developer. If most developers make the same mistake, it's the fault of the language.

- **Re:Oh please (Score:5, Interesting)**
by Dogtanian (588974) on Tuesday February 21, 2017 @06:05PM (#53908249) Homepage
Any language where the default equality comparison operator is *true* given two string-type variables with values "0E54321" and "0E12345" is not a cryptographically secure language. In fact there is a nonzero chance of the default equality operator returning true between two different MD5 or SHA256 hashes if they happen to fall into a hexadecimal form that is all digits except for one E or F.
Technically, that (in itself) doesn't necessarily mean that the built-in cryptography nor the language itself are inherently insecure. In theory, that is, provided you understand the language and use it correctly.

And that's the problem. Because in practice, PHP's design philosophy of trying to be clever- often too clever by half- when it comes to comparisons, equality, automatic coercion, data types, etc. etc. too often gives unpredictable and unexpected results from people who weren't aware of that behaviour.

You absolutely do *not* want any risk of this happening when you're designing a system that has to be secure. You want boringly explicit and utterly predictable data and type handling.

My prediction is that far, *far* more security holes will be down to bugs caused by unforeseen subtle aspects- i.e. pitfalls- of PHP's type handling and equality behaviour (etc.) in the apps using it rather than bugs in the cryptographic module itself.

PHP being a language more favoured by inexperienced users, this is likely to be made far worse. Expect lots of newbies with misplaced confidence designing what they think are "secure" apps that are in fact full of holes- either because they've misused or misunderstood the cryptographic module, or because they've overlooked some basic aspect of computer security elsewhere (e.g. failure to parse input securely) that makes the use of cryptography irrelevant.

And those are the sorts of mistakes newbies would make when using any language- with PHP's language design issues on top of that, it has the potential to be far worse.

So, yeah. I trust that the module will be secure. The main problems- I guarantee- will be caused by caused by overlooked (or not known about) aspects of PHP's too-clever-by-half data handling (in client apps using it) leading to exploitable holes, and by the fact that too many of PHP's newbie-skewing userbase will overconfidently assume it makes their apps foolproof while using it incorrectly and ignoring security holes elsewhere that make it redundant.
Reply to This | Parent | Share
twitter facebook linkedin
Flag as Inappropriate

- **Re: (Score:1)**
by Lisandro (799651)
Agreed. The problem is not the library - PHP is *intrinsically* insecure. I'm not aware of any other language out there that can reliably be made to segfault [php.net] and whose developers argue such behavior is working as intended.

- **Re: (Score:2)**
by Ksevio (865461)
I can get a segfault pretty reliably in C with this code:
`char *s = "hello world"; *s = 'H';`

- **Re: (Score:3)**
by Lisandro (799651)
I think you meant `s = 'H';` there....
But again, this is to be expected. C is a low level language without memory management; PHP is neither. There is no sane reason why it is *supposed* to segfault (according to the developers, "infinite recursion crashes") and i'm not aware of any other high level language which does the same. How hard is to add a recursion counter?

- **Re: (Score:2)**
by goose-incarnated (1145029)
I think you meant `s = 'H';` there....
No, he is correct that `*s = 'H'` can segfault. You aren't allowed to modify string literals, only char arrays.

- **Re: (Score:2)**
by Lisandro (799651)
Oh, it **can**. It just won't do it reliably. Setting a string pointer to 0x72 will certainly do.

- **Re: (Score:2)**
by goose-incarnated (1145029)
Oh, it **can**. It just won't do it reliably. Setting a string pointer to 0x72 will certainly do.
It can. It might not though. It depends, actually, on whether it is ever deferenced or not. The original `*s = 'H'` can crash without ever needing any more derefencing. Your correction can (not 'will', but 'can') crash only if it is dereferenced.
Setting it is not enough, you *have* to dereference it as well. Although now that I think about it, depending on the platform, simply dereferencing it may still not be enough.
For example, I just tried it now, verbatim, in the OS kernel I hobby-developed and it worked ju

- **Re: (Score:2)**
by goose-incarnated (1145029)
Dereferencing the virtual address 0x72 is guaranteed to crash an application when the OS is designed never to map the first (usually 4K) addresses. Accessing that memory causes a page fault, the OS searches the page table for the corresponding page mapping, fails to find a valid mapping, so you get a segmentation fault.
This has nothing to do with C though. C says the behavior is "undefined" if you set a pointer to a location that is not used to store a C object.
We've already agreed that it's undefined behaviour. What we're arguing are the circumstances that this particular invocation of UB would result in a crash. Just because something is UB doesn't mean that it consistently crashes at the point of UB.

- **Re: (Score:2)**
by gweihir (88907)
Because modifying string _constants_ is a bad idea?

- **Re: (Score:2)**
by TheRaven64 (641858)
The type of "hello world" is const char *, so your compiler should warn that you're dropping the const in an implicit cast (and if you're a sensible person and compile with -Werror, then your compiler will reject the code). You can get the behaviour that you want with:
`const char s[] = "hello world";`
This will copy the contents of the string literal into a mutable array. If you write this at the global scope, the copy will be done at compile time, so you'll end up with the string in the data section, not

- **Re: (Score:2)**
by gweihir (88907)
C is not an interpreted language with memory safety.Your example does not apply and it is completely clueless in addition.

- **Re: (Score:2)**
by angel'o'sphere (80593)
But that depends on many things like operating system and Compiler, and I would not wonder if you forgot a 'const' to force the crash ;D

- **Re: (Score:2)**
by mysticgoat (582871)
To repeat: quit blaming the language for the lack of skills of the programmer.
Programmers who are incompetent in PHP should not attempt to build secure systems with it. Those who do attempt to do so should really not be involved in making secure systems in any language, since they have demonstrated that they do not know how to assess the limitations of their tools, let alone work properly within those limitations.
It is a poor craftsman who blames his tools for his shoddy workmanship.

- **Re: (Score:3)**
by gweihir (88907)
It is a poor craftsman that uses shoddy tools in the first place. Selecting good tools is a core skill for any craftsman. Those that do not have it will never amount to anything.

- **Re: (Score:2)**
by gweihir (88907)
And there we have the core problem: Clueless language designers with big egos. A segfault is like scramming a reactor. It is an absolute least resort, it may do significant damage, it is always a sign of something having gone terribly wrong and it should never, ever happen in normal operation.

- **Re: (Score:2)**
by Lisandro (799651)
You do know that "feature" is present on all PHP versions up to the 7.x line, right?

- **Re: (Score:1)**
by Anonymous Coward
Came for the PHP haters, was not disappointed.

- **Dynamic Type Comparing [Re:Oh please] (Score:1)**
by Tablizer (95088)
In my opinion, dynamic languages should require, or at least encourage, one to specify what comparison type to use rather than rely on parameter (operand) analysis. The hard part is coming up with a nice syntax for such. I've had various discussion groups consider different suggestions, and found no consensus.
In the shorter term, one can roll their own functions and hope staff coders follow along. Example:
```
if (strCompare(a, '>', b)) {...}
if (numCompare(a, '<=', b)) {...}
```

- **Re: (Score:2)**
by gweihir (88907)
Very much this. PHP is a language for idiots that have no clue they are idiots. It is designed by the same class of people, as it ignores most, if not all, principles of good language design.
Without a large group of people that consider themselves "coders", but are anything but (unless you include the lowest level of amateur-activity) a language like PHP would never have stood any chance. It is just too much of an insult to anybody with real skills in the area.

- **Re: (Score:1)**
by rp (29053)
Please note that the article claims nothing of the kind.

SubjectsSuck (Score:5, Informative)
by aardvarkjoe (156801) on Tuesday February 21, 2017 @05:09PM (#53907935)
Arciszewski also says that PHP is actually "the first" programming language to support a "modern" cryptography library in its core, despite Erlang and Go including similar libraries, which he claims are not as powerful and up-to-date as PHP's upcoming Libsodium implementation.
So it's the first to support a modern cryptography library, as long as you define "modern" to mean "the one that we're using."
It's easy to be first to do something if you place enough arbitrary restrictions on what that something is.
Reply to This | Share
twitter facebook linkedin
Flag as Inappropriate

- **Re:SubjectsSuck (Score:4, Insightful)**
by NotInHere (3654617) on Tuesday February 21, 2017 @05:13PM (#53907953)
"Modern" is for CS people like "Alternative facts" is for politicians.
Reply to This | Parent | Share
twitter facebook linkedin
Flag as Inappropriate

- **Re:SubjectsSuck (Score:4, Interesting)**
by MightyMartian (840721) on Tuesday February 21, 2017 @05:18PM (#53907979) Journal
I don't even understand the point of the claim. So the interpreter has a baked-in crypto library? And how is that different than simply #including a crypto library, which has the added bonus that you can pick any number of crypto libraries.
Reply to This | Parent | Share
twitter facebook linkedin
Flag as Inappropriate

- **Would you prefer an interpreted crypto library? (Score:4, Insightful)**
by tepples (727027) <tepples.gmail@com> on Tuesday February 21, 2017 @06:15PM (#53908291) Homepage Journal
And how is that different than simply #including a crypto library, which has the added bonus that you can pick any number of crypto libraries.
I can see three ways to proceed:
A built-in crypto library
This runs at full speed and is available by default to the shared hosting customer.
An add-on crypto library compiled to native code and distributed as a PHP extension
This runs at full speed but requires the shared hosting customer to convince the hosting provider to install it.
An add-on crypto library written in pure PHP
This is available by default to the shared hosting customer but can run unacceptably slowly due to interpreter overhead.
Reply to This | Parent | Share
twitter facebook linkedin
Flag as Inappropriate

- **Re: (Score:2, Insightful)**

by Anonymous Coward
Libsodium is an extended fork of Daniel J. Bernstein's [wikipedia.org] original NaCl [cr.yp.to] project (not to be confused with Google Native Client), which is a cryptography library developed with the overarching aim of simplifying (and improving the implementation-level safety of) the practical use of strong cryptographic constructs. The "big idea" behind NaCl was to abstract away many of the low-level choices and technical details associated with various cryptographic primitives in favor of more "generic" interfaces, utilizing im

- **Re: SubjectsSuck (Score:2)**
by Sledgy (133446)
It's different in that if a vulnerability is discovered in the underlaying library an new release of PHP is required to fix it. Note you will need to convince the hosting provider/infrastructure team to test and deploy the latest release. This is the reason the cryptography library for Python (and requests) don't want to be made part of the core so they can be agile and respond quickly to security problems.

- **Re: (Score:2)**
by Waccoon (1186667)
Well, it certainly makes things easier for us hobby programmers that make redistributable projects designed to run on shared servers. I can't count on people installing a framework or keeping a library up to date if they have no admin access to the server. A lot of PHP projects, like Wordpress, still aren't aimed at serious, large-scale enterprises. Don't assume the owners have shell access with enough permission to install dependencies.
I started working with PHP over a decade ago because it had a graphi

- **Re: (Score:1)**
by SCY.tSCc. (514610)
It's easy to be first to do something if you place enough arbitrary restrictions on what that something is.
Hey, you're the first user in this thread whose user id starts with 15680 to say THAT.

- **Re: (Score:2)**
by Darinbob (1142669)
Tomorrow there may be a new security library and the first language that uses it will be the "first" language to use a "modern" security library.
Pity the person who's been campaigning to include SSL into a language only to be told "we're deferring this because it's no longer modern enough so we will continue providing no security library."

- **Re: (Score:2)**
by bluefoxlucid (723572)
More ridiculous is the claim that including crypto will force WordPress to implement better security. WordPress can just ignore this; and getting hacked by shitty REST API authentication verification isn't fixed by pouring on more crypto sauce.
This guy is a crypto nerd who thinks crypto solves all problems. It doesn't. He probably has databases with columns (UserID, UserName, CryptedPassword, AESKey) so the password is AES-encrypted with an individual key per-user.

- **Re: (Score:3)**
by Chuck Chunder (21021)
What are the best alternatives to NACL for cryptographic primitives?
I think the point is "first" is a weird word to use when you are talking about "modern" as "modern" changes with time.

OpenSSL or mcrypt or whatever else you might point to were "modern" when they were "first" used, even if they aren't "modern" any more.

"Only" might be a better choice if you are talking about the current time.

- **So they'll be the first to do it wrong? (Score:4, Interesting)**
by OverlordQ (264228) on Tuesday February 21, 2017 @05:14PM (#53907955) Journal
I'll stick to every other language that has had libsodium bindings for a while now.
Reply to This Share
twitter facebook linkedin
Flag as Inappropriate

- **Re:So they'll be the first to do it wrong? (Score:5, Funny)**
by c (8461) <beauregardcp@gmail.com> on Tuesday February 21, 2017 @05:17PM (#53907971)
I'll stick to every other language that has had libsodium bindings for a while now.
I'm just waiting for them to release the libsodium bindings for C...
Reply to This Parent Share
twitter facebook linkedin
Flag as Inappropriate

- **Re: (Score:2)**
by thegarbz (1787294)
The only way to really get libsodium to bind with C is to use Google's Native Client environment.

- **Re: (Score:2)**
by gweihir (88907)
They probably only care to be the "first" at something.

- **Re: (Score:2)**
by AlphaBro (2809233)
Hilarious this was modded down to troll. It's a legitimate request.

- **libsodium is a C library (Score:2)**
by adam.voss (1854938)
For those who don't know, libsodium is a C library [github.com] that PHP will be utilizing. It is not a PHP library.

- **Re: (Score:1)**
by Anonymous Coward
That's actually a good point. PHP is a bit unlike most other "interpreted" languages in that there are two kinds of libraries. There are the mods, written in native C, and included in the PHP+Apache runtime environment. That reflects PHP's basic structure as more-or-less a fancy wrapper around a bunch of C libraries. Then there are libraries written in PHP, such as the stuff in the PEAR libraries.
AFAIK (but I'm ignorant on this topic), Java, Javascript/node.js, etc. don't have a way to include C libraries i

- **Re: libsodium is a C library (Score:1)**
by p91paul (4513273)
Java has support for calling C (actually, any compiled library) through JNI (Java Native Interface)

- **Too little... too late... (Score:3)**
by creimer (824291) on Tuesday February 21, 2017 @05:20PM (#53908001) Homepage
I got tired of script kiddies banging down my PHP/MySQL servers. I'm using Pelican (Python) to convert my websites into static websites. With nothing to hack, script kiddies go away.
Reply to This Share
twitter facebook linkedin
Flag as Inappropriate

- **Re: (Score:2)**
by tepples (727027)
I'm using Pelican (Python) to convert my websites into static websites. With nothing to hack, script kiddies go away.
How do you edit your Pelican-powered website while away from your home PC? Skiddies can hack through that.

- **Re: (Score:2)**
by creimer (824291)
How do you edit your Pelican-powered website while away from your home PC? Skiddies can hack through that.
I don't allow outside access to my file server at home. From my file server I can make whatever changes needed to the website, run pelican to generate the static files, and rsync the output directory to the hosting server. Since I don't use PHP or MySQL, the script kiddies have no attack vector into my website.

- **Re: (Score:2)**
by tepples (727027)
I don't allow outside access to my file server at home. From my file server I can make whatever changes needed to the website
If you get an idea for an update while away from home, particularly for an extended period, what do you do with that idea?

- **Re: (Score:3)**
by creimer (824291)
If you get an idea for an update while away from home, particularly for an extended period, what do you do with that idea?
I write it down in my dead tree notebook.

- **Re: (Score:2)**
by thegarbz (1787294)
I don't allow outside access to my file server at home. From my file server I can make whatever changes needed to the website
If you get an idea for an update while away from home, particularly for an extended period, what do you do with that idea?
Sorry but this is a silly thread. Security is an inconvenience and you have to live with it. That said your problem is not solvable:
1) write it down.
2) email it to myself.
3) ssh.
4) outright VPN.
Me I take the last option. There's no dynamic content accessible on my site which isn't behind at the least HTTP auth. There's no dynamic content on my website which takes user input that doesn't also limit it's scope to the local network.
And despite all of these restrictions, content is still easier to update than i

- **Ahhh PHP.... (Score:2)**
by cmdrbuzz (681767)
PHP, the "Speak 'n Spell" of programming languages.... More marketing fluff.

- **Re: (Score:2)**
by AlphaBro (2809233)
Unless it's a buffer over-read, in which case there might not be a 'BZZZZT!' You know, like Heartbleed.

- **BULLSHIT! (Score:3)**
by allo (1728082) on Tuesday February 21, 2017 @05:24PM (#53908017)
BULLSHIT! BULLSHIT! BULLSHIT!
PHP is one of the programming languages, which load all stuff into the core (which can be quite a disadvantage), but other languages use a library by a single include. So what?
And even php has it into a .so file, which can be loaded, but isn't required to be used. So the "core" is relative as well. Actually its a bundled module.
Reply to This Share
twitter facebook linkedin
Flag as Inappropriate

- **Re: (Score:1)**
by rp (29053)
Bullshit. PHP doesn't load everything into its core: it has a well-developed system for managing optional extensions (PEAR). "Core" extensions are the ones that come with PHP.. This is pretty standard terminology. You just can't be bothered to look it up. Then why reply at all?

- **Re: (Score:2)**
by allo (1728082)
you're confusing php modules with php extensions. Pear installs php scripts, not modules. The modules are stuff like libgd for graphics and so on, coming as shared library (.dll or .so).

- **Re: (Score:2)**
by allo (1728082)
> but I don't recall Java or Javascript (for example) having any equivalent method for including a .so file into their runtime either permanently or on demand.
Javascript isn't the best example, as it doesn't have any good standard library (which leads to the whole npm fuckup).
Java can load binary modules, but much stuff is written in java and performs good enough.
But the actual point is, that you link against a libsodium binding and php does it as well. They have build some lib, which then provides funct

- **Perhaps instead of building everything and ..... (Score:2)**
by aix tom (902140)
.... a kitchen sink into the core, they could have instead done a *sane* way to include additional modules.
Perl and Python for example have no problem loading user-specific or script-specific modules, not like the "system wide or nothing" approach of PHP. (which of course doesn't work with shared hosting.)

- **Other languages did this first (Score:4, Interesting)**
by MobyDisk (75490) on Tuesday February 21, 2017 @05:32PM (#53908063) Homepage
I remember when Java was the first language to do this. Shortly after that, C# was the first language to do this. Now PHP is the first language to do this. So who will be the next one to do it first?
Reply to This Share
twitter facebook linkedin
Flag as Inappropriate

- **I'm not sure this is a good idea (Score:2)**
by jandrese (485)
I'm torn on the idea of having one particular crypto implementation having first class citizen status in the language. It should help adoption and alleviate deployment headaches, but if that library turns out to have problems or just becomes obsolete it's even more of a hassle to work around it. Crypto algorithms are unusual in computer science in that they come with use-by dates. Most algorithms are timeless, but crypto changes constantly. What are the odds that in 5 years this becomes "that thing you

- **Re: (Score:2)**
by gweihir (88907)
This is PHP. The language is saturated with bad decisions. This is just one more of those.

- **Gotta love PHP ... (Score:4, Funny)**
by Qbertino (265505) <moiraNO@SPAMmodparlor.com> on Tuesday February 21, 2017 @06:04PM (#53908235)
I'm smiling while I read this.
Every single bit of this news is sooo PHP and one of the reasons this awkward mess of a PL is so successful. [slashdot.org]
They find something new or something they need and bolt it on. Just like that. End of story. A vote on the core team, a little coding and ***BAM*** PHP has a new inner API function with what has to be the most over-the-top all-out-PHP-style name for an inner API function ever - `sodium_crypto_box_keypair_from_secretkey_and_publickey($ecdh_secret, $ecdh_public);` (seriously, this is no joke).
Totally LOL. Takes the cake for inner function names ten times over, even by PHP standards, which is quite a stunt. And right away PHP has up-to-date hard crypto that even a simpleton can use.
You have to hand it to the PHP crew - they actually get shit done, no matter what. :-)
Reply to This Share
twitter facebook linkedin
Flag as Inappropriate

- **Re: (Score:2)**
by phantomfive (622387)
You have to hand it to the PHP crew - they actually get shit done, no matter what. :-)
They should make that their motto: "Getting shit done, no matter what!"

- **Monte beat PHP by a year! (Score:3)**
by MostAwesomeDude (980382) on Tuesday February 21, 2017 @06:04PM (#53908237) Homepage
My beloved Monte https://monte.rtfd.org/ [rtfd.org] beat PHP to this by a wide stretch. While it's true that PHP is a big established language, that doesn't mean that they get to claim sudden leaps in innovation which didn't happen. I've tweeted at the author of the blog post https://twitter.com/corbinsimpson/status/834175224736157696 [twitter.com] with timestamped commits from the Monte codebase.
Reply to This Share
twitter facebook linkedin
Flag as Inappropriate

- **So, you're embedding libsoduim... (Score:4, Interesting)**
by Lisandro (799651) on Tuesday February 21, 2017 @06:06PM (#53908251)
...which effectively prevents me from updating it. Great choice for a security library guys.
Reply to This Share
twitter facebook linkedin
Flag as Inappropriate

- **Re: (Score:1)**
by Anonymous Coward
Most software is (or should be) subject to maintenance updates for various reasons. Embedding libsodium in an application simply shifts the distribution point for updates. If you're using software (PHP in this case) distributed as a package in a common Linux or BSD distribution, you'll have the ability to install updates/fixes whenever the distribution's package maintainers make them available from new or patched upstream sources. If you're compiling from source, you'll have the option of tracking updates a

- **Re: (Score:2)**
by Lisandro (799651)
Sorry, that is not enough. There's a vast difference from updating a crypto library (which can happen, f.ex, due to security updates) to updating *the whole damn language*, which can have system-wide implications.

- **Re: (Score:2)**
by Lisandro (799651)
I understand what you're saying; my point is that the problem is **not** the release model.
First off, i'm not aware of any languages which do rolling releases - PHP certainly doesn't - and with good reason. The problem is that PHP makes a security component part of the language itself and provides no other way of updating it; a libsodium update becomes now a PHP update, which means you're at the whim and mercy of the language creators when it comes to managing it.
This is specially annoying in PHP, where languag

- **Re: (Score:2)**
by Lisandro (799651)
That's quite more than "a bit of thinkering", specially compared with pretty much any other interpreted language in the *nix word, which allow multiple language/library versions on a single system in a much saner way.

- **Every time I see a PHP job ad I think (Score:1)**
by FryingLizard (512858)
"...man, you guys must have some serious technical debt"
I built a startup's entire stack on PHP back in the 2003-2006 time, now I look back and SMH at the foolishness. If you want a quick'n'weakly-typed language (which I often do), Python beats the crap out of PHP, as well as being ten times more readable.

- **The crypto is the easy part of this (Score:3)**
by thogard (43403) on Tuesday February 21, 2017 @07:34PM (#53908595) Homepage
The cryptography algorithms are the easy part. The vary hard part is protecting keys so I hope someone provides plenty of examples of how to do that properly. I hope they don't go down the Java route of showing how to use the functions without proper key management.
Reply to This Share
twitter facebook linkedin
Flag as Inappropriate

- **And it'll be a shitshow because of course it is (Score:5, Funny)**
by Just Some Guy (3352) <kirk+slashdot@strauser.com> on Tuesday February 21, 2017 @08:03PM (#53908697) Homepage Journal
Sneak preview of the API:
```
crypto_really_encode(plaintext, algorithm); // Simplest
crypto_really_encode(plaintext, mode, algorithm); // Next arg goes in the middle
crypto_really_encode(block_size, plaintext, algorithm, mode); // Switch it up yo lol
```
...where `AES` will somehow be a valid value for both `mode` and `algorithm` (which will silently override to "NULL" if `plaintext` starts with a zero or the letter "p").
Reply to This Share
twitter facebook linkedin
Flag as Inappropriate

- **Re: (Score:2)**
by AlphaBro (2809233)
Lots of gold in this thread, but this one should be a 5. The PHP API design decisions are the shittiest I've ever seen in my life.

- **Oh LOL! PHP! (Score:1)**
by dschiptsov (4126095)
FYI Golang has a "modern" cryptography library in its core. PHP in 21th century. What a lunacy.

- **Re: (Score:2)**
by rubycodez (864176)
guess back in the late 80s ANSI weren't smart enough to include block cipher libraries we were calling from FORTRAN and C into the languages. pfft, php never disappoints, it's like the QBASIC of the 21st century

- **Re: (Score:2)**
by AlphaBro (2809233)
Yes, but you see, Microsoft didn't statically link all the crypto stuff into a monolithic binary. Because, uh... I guess loading a bunch of unused stuff into memory is newsworthy.

- **Re: (Score:2)**
by gweihir (88907)
You have to admit that they are consistent at the bad decisions though. Making bad decisions seems to be their most important guiding principle.

Related Links Top of the: day, week, month.
- 674 comments Ask Slashdot: What Are Some Bad Programming Ideas That Work?
- 600 comments Developer Argues For 'Forgotten Code Constructs' Like GOTO and Eval
- 531 comments Are Flawed Languages Creating Bad Software?
- 523 comments Linus Torvalds In Sweary Rant About Punctuation In Kernel Comments
- 501 comments Will The Death of the PC Bring 'An End To Openness'?

next
UPS Develops 'Rolling Warehouse' System In Which Drones Are Launched From Atop Trucks
38 comments
previous
Health Apps Could Be Doing More Harm Than Good, Warn Scientists
86 comments

Slashdot
Post
Get 75 More Comments
100 of 174 loaded
Submit Story

I am here by the will of the people and I won't leave until I get my raincoat back. - a slogan of the anarchists in Richard Kadrey's "Metrophage"

FAQ
Story Archive
Hall of Fame
Advertising
Terms
Privacy
Cookie Preferences
Opt Out Choices
About
Feedback
Mobile View
Blog

Close
Slashdot

Working...