Two-way password encryption without ssl

Asked 16 years, 3 months ago Modified 2 years, 6 months ago

Viewed 17k times

14

I am using the basic-auth twitter API (no longer available) to integrate twitter with my blog's commenting system. The problem with this and many other web APIs out there is that they require the user's username and password to do anything useful. I don't want to deal with the hassle and cost of installing a SSL certificate, but I also don't want passwords passed over the wire in clear text.

I guess my general question is: **How can I send sensitive data over an insecure channel?**

This is my current solution and I'd like to know if there are any holes in it:

1. Generate a random key on the server (I'm using php).
2. Save the key in a session and also output the key in a javascript variable.
3. On form submit, use Triple DES in javascript with the key to encrypt the password.
4. On the server, decrypt the password using the key from the session and then destroy the session.

The end result is that only the encrypted password is sent over the wire and the key is only used once and never sent with the password. Problem solved?

javascript security encryption passwords

Share

Improve this question

Follow

edited Feb 13, 2013 at 16:21

asked Sep 2, 2008 at 4:49

dsims

1,322 ● 1 ● 15 ● 24

2 You can't do it. SSL is expensive because a third party that all browsers trust has verified that your server is the real one for a given domain name. Encryption is useless when an attacker can force users into thinking their server is the host for your domain and this attack takes a split second to perform on a wifi or ethernet network (library, school, office, etc). See the back catalogue of the security now podcast to learn all the issues, you'll soon decide it's easier and cheaper to buy an SSL cert from someone cheap. – Abhi Beckert Jul 22, 2013 at 18:31

13 Answers

Sorted by: Highest score (default)

34

1. Generate a random key on the server (I'm using php).

2. Save the key in a session and also output the key in a javascript variable.
3. On form submit, use Triple DES in javascript with the key to encrypt the password.

This avoids sending the password in the clear over the wire, but it requires you to send the key in the clear over the wire, which would allow anyone eavesdropping to decode the password.

It's been said before and I'll say it again: don't try to make up your own cryptographic protocols! There are established protocols out there for this kind of thing that have been created, peer reviewed, beat on, hacked on, poked and prodded by professionals, **use them!** No one person is going to be able to come up with something better than the entire cryptographic and security community working together.

Share Improve this answer

Follow

answered Sep 2, 2008 at 5:03

Chris Upchurch

15.5k ● 6 ● 52 ● 66

9 +1 for "don't try to make up your own cryptographic protocols!" :D – Jalal Apr 11, 2011 at 9:35

10

Your method has a flaw - if someone were to intercept the transmission of the key to the user and the user's

encrypted reply they could decrypt the reply and obtain the username/password of the user.

However, there is a way to securely send information over an unsecure medium so long as the information is not capable of being modified in transit known as the Diffie-Hellman algorithm. Basically two parties are able to compute the shared key used to encrypt the data based on their conversations - yet an observer does not have enough information to deduce the key.

Setting up the conversation between the client and the server can be tricky though, and much more time consuming than simply applying SSL to your site. You don't even have to pay for it - you can generate a self-signed certificate that provides the necessary encryption. This won't protect against man-in-the-middle attacks, but neither will the Diffie-Hellman algorithm.

Share Improve this answer Follow

edited Sep 2, 2008 at 5:06

answered Sep 2, 2008 at 4:55

Kyle Cronin

79k ● 45 ● 151 ● 167

"so long as the information is not capable of being modified in transit" - that is almost impossible to achieve over a http connection without SSL. It is trivial to modify the information in transit. – Abhi Beckert Jul 22, 2013 at 18:34

6

You don't have to have a certificate on your server; it's up to the client whether they are willing to talk to an unauthenticated server. Key agreement can still be performed to establish a private channel. It wouldn't be safe to send private credentials to an unauthenticated server though, which is why you don't see SSL used this way in practice.

To answer your general question: **you just send it.** I think your *real* general question is: *"How do I send sensitive data over an insecure channel—and keep it secure?"* **You can't.**

It sounds like you've decided that security isn't worth the $10–20 per month a certificate would cost, and to protect Twitter passwords, that's probably true. So, why spend time to provide the illusion of security? Just make it clear to your users that their password will be sent in the clear and let them make their own choice.

Share Improve this answer

Follow

answered Sep 2, 2008 at 5:25

erickson

269k ● 59 ● 401 ● 497

2

So how is this any more secure? Even though you might have secured browser<>your server, what about the rest of the Internet (your server<>twitter)?

IMHO, it's unacceptable to ask for a username and password of another service and expect people to enter

that. And if you care that much - don't integrate them until they get their act straight and re-enable OAuth. (They supported it for a while, but disabled it a few months ago.)

In the mean time, why not offer OpenID? Every Google, Yahoo!, VOX etc. account has one. People might not be aware of it but chances are really, really high that they already have OpenID. Check this list to see what I mean.

Share Improve this answer

Follow

answered Sep 2, 2008 at 5:06

Till

22.4k ● 4 ● 60 ● 89

> Please can you explain the first sentence of your second paragraph? As long as you trust the website asking you your user name and your password, why would it be unacceptable? It can reduce password fatigue and if it is safely implemented (HTTP Secure, SSL/TLS, ...), it can be very useful. Moreover, this is typically what StackOverflow does by allowing us to use a Google account for example. Finally, there are still tons of email providers with no support of OpenID and OpenID Connect. – gouessej Jan 22, 2016 at 13:36

2

When the key is sent between the client and the server it is clear text and subject to interception. Combine that with the encrypted text of the password and the password is decrypted.

Diffie-Hellman is a good solution. If you only need to authenticate them, and not actually transmit the password (because the password is already stored on the server)

then you can use HTTP Digest Authentication, or some variation there of.

Share Improve this answer Follow

answered Sep 2, 2008 at 5:12

Jim McKeeth

38.7k ● 25 ● 124 ● 199

2

APIs and OAuth

Firstly, as others have said, you shouldn't be using a user's password to access the API, you should be getting an OAuth token. This will allow you to act on that user's behalf without needing their password. This is a common approach used by many APIs.

Key Exchange

If you need to solve the more general problem of exchanging information over insecure connections, there are several key exchange protocols as mentioned by other answers.

In general key exchange algorithms are secure from eavesdroppers, but because they do not authenticate the identity of the users, they are vulnerable to man-in-the-middle attacks.

From the Wikipedia page on Diffie Hellman:

> In the original description, the Diffie–Hellman exchange by itself does not provide

> authentication of the communicating parties and is thus vulnerable to a man-in-the-middle attack. A person in the middle may establish two distinct Diffie–Hellman key exchanges, one with Alice and the other with Bob, effectively masquerading as Alice to Bob, and vice versa, allowing the attacker to decrypt (and read or store) then re-encrypt the messages passed between them. A method to authenticate the communicating parties to each other is generally needed to prevent this type of attack. Variants of Diffie-Hellman, such as STS, may be used instead to avoid these types of attacks.

Even STS is insecure in some cases where an attacker is able to insert their own identity (signing key) in place of either the sender or receiver.

Identity and Authentication

This is exactly the problem SSL is designed to solve, by establishing a hierarchy of 'trusted' signing authorities which have *in theory* verified who owns a domain name, etc, someone connecting to a website can verify that they are indeed communicating with that domain's server, and not with a man-in-the-middle who has placed themselves in between.

You can create a self-signed certificate which will provide the necessary configuration to encrypt the connection, but will not protect you from man in the middle attacks for

the same reason that unauthenticated Diffie-Hellman key exchange will not.

You can get free SSL certificates valid for 1 year from https://www.startssl.com/ - I use them for my personal sites. They're not quite as 'trusted' whatever that means, since they only do automatic checks on people who apply for one, but it's free. There are also services which cost very little (£10/year from 123-Reg in the UK).

Share Improve this answer

Follow

answered Feb 13, 2013 at 12:37

Kothar

6,629 ● 3 ● 35 ● 44

1

I've implemented a different approach

1. Server: user name and password-hash stored in the database
2. Server: send a challenge with the form to request the password, store it in the session with a timestamp and the client's IP address
3. Client: hash the password, concat challenge|username|passwordhash, hash it again and post it to the server
4. Server: verify timestamp, IP, do the same concatenation/hashing and compare it

This applies to a password transmission. Using it for data means using the final hash as the encryption key for the

plain text and generating a random initialization vector transmitted with the cipher text to the server.

Any comments on this?

Share Improve this answer

Follow

answered Sep 24, 2008 at 16:25

Oli

1,782 ● 3 ● 19 ● 22

1

The problem with client-side javascript security is that the attacker can modify the javascript in transit to a simple {return input;} thereby rendering your elaborate security moot. Solution: use browser-provided (ie. not transmitted) RSA. From what I know, not available yet.

Share Improve this answer

Follow

answered Apr 25, 2016 at 10:45

Kadigan.KSB

11 ● 1

0

> How can I send sensitive data over an insecure channel

With a pre-shared secret key. This is what you attempt in your suggested solution, but you can't send that key over the insecure channel. Someone mentioned DH, which will help you negotiate a key. But the other part of what SSL does is provide authentication, to prevent man-in-the-middle attacks so that the client knows they are

negotiating a key with the person they intend to communicate with.

Chris Upchurch's advice is really the only good answer there is for 99.99% of engineers - don't do it. Let someone else do it and use their solution (like the guys who wrote the SSL client/server).

I think the ideal solution here would be to get Twitter to support OpenID and then use that.

Share Improve this answer

Follow

answered Oct 31, 2008 at 15:56

bmm6o

6,485 ● 3 ● 31 ● 57

0

An ssl certificate that is self-signed doesn't cost money. For a free twitter service, that is probably just fine for users.

Share Improve this answer

Follow

answered Oct 31, 2008 at 16:09

Marcin

49.8k ● 18 ● 132 ● 206

0

TO OLI

In your approch for example i'm in the same subnet with same router, so i get the same ip as my collegues in my work. I open same url in browser, so server generates the timestamp with same ip, then i use tcp/ip dump to sniff

the hashed or non hashed password from my collegues connection. I can sniff everything he sends. So i have all hashes from his form also you have timestamp(my) and same ip. So i send everything using post tool and hey i'm loggen in.

Share Improve this answer

Follow

answered Sep 25, 2010 at 18:14

bad_man

1

0

If you don't want to use SSL, why not try some other protocol, such as kerberos?

A basic overview is here:
http://www.kerberos.org/software/tutorial.html

Or if you want to go somewhat more in depth, see
http://www.hitmill.com/computers/kerberos.html

Share Improve this answer

Follow

edited Nov 29, 2011 at 1:14

user47589

answered Nov 29, 2011 at 0:55

Jack

1

1 IMO, this is more hassle than just installing an SSL certificate. – user47589 Nov 29, 2011 at 1:15

0

I have a similar issue(wanting to encrypt data in forms without paying for an ssl certificate) so I did some hunting and found this project: http://www.jcryption.org/

I haven't used it yet but it looks easy to implement and thought I'd share it here in-case anyone else is looking for something like it and finds themselves on this page like I did.

Share Improve this answer

Follow

answered Apr 19, 2012 at 21:26

Dsyko

1,514 ● 15 ● 27

So now that I've used the project I can say it's awesome and really easy to integrate into a project if you are using PHP. Of course it doesn't really give you 100% protection, see here But if you're just looking for a way to add more protection that sending form elements in clear-test Posts... this is a great option. – Dsyko Apr 26, 2012 at 12:01