Re: iOS 11 HTTPS network request failure with self-signed cert

37998 Views 17 Replies Latest reply on Jun 9, 2019 11:42 PM by Piaojin

Level 1
(0 points)

hanxuanfromtoronto
Sep 22, 2017 1:56 PM

For testing purpose, we use a self-signed cert, loaded as a bundle resource for network requests.
It works fine on Xcode8 + iOS10 devices and simulators.
On Xcode9 and iOS 11, however, prompts with error as:

```
----------------
<APIClient.swift> call [#77]
Error Domain=NSURLErrorDomain Code=-1200 "An SSL error has occurred and a secure
connection to the server cannot be made." UserInfo=
{NSURLErrorFailingURLPeerTrustErrorKey=<SecTrustRef: 0x1c0111f70>,
NSLocalizedRecoverySuggestion=Would you like to connect to the server anyway?,
_kCFStreamErrorDomainKey=3, _kCFStreamErrorCodeKey=-9802,
NSErrorPeerCertificateChainKey=(
    "<cert(0x1070fb600) s: *.😎 i:🙂>",
    "<cert(0x1070fce00) s:🙂 i: 2048 offline root>",
    "<cert(0x1070fd600) s: 2048 offline root i: 2048 offline root>"
), NSUnderlyingError=0x1c465e180 {Error Domain=kCFErrorDomainCFNetwork Code=-1200
"(null)" UserInfo={_kCFStreamPropertySSLClientCertificateState=0,
kCFStreamPropertySSLPeerTrust=<SecTrustRef: 0x1c0111f70>,
_kCFNetworkCFStreamSSLErrorOriginalValue=-9802, _kCFStreamErrorDomainKey=3,
_kCFStreamErrorCodeKey=-9802, kCFStreamPropertySSLPeerCertificates=(
    "<cert(0x1070fb600) s: *.😎 i:🙂>",
    "<cert(0x1070fce00) s:😎 i: 2048 offline root>",
    "<cert(0x1070fd600) s: 2048 offline root i: 2048 offline root>"
)}}, NSLocalizedDescription=An SSL error has occurred and a secure connection to the
server cannot be made., NSErrorFailingURLKey=https:/
----------------
```

It can be resolved by adding:

```
    <key>NSAppTransportSecurity</key>
    <dict>
        <key>NSAllowsArbitraryLoads</key>
        <true/>
    </dict>
```

I wonder if the security policy is changed, like self-signed cert is totally prohibited on iOS 11?

If yes, is there a link/documentation reference?

Tags: ats, network, https, ssl cert, self-signed, ios 11

Apple Staff
(13,265 points)

eskimo
Sep 27, 2017 2:09 AM
(in response to hanxuanfromtoronto)

> I wonder if the security policy is changed

iOS 11 does have a number of security policy changes related to HTTPS server trust evaluation.

> like self-signed cert is totally prohibited on iOS 11?

But this is not one of them.
For the details you should watch WWDC 2017 Session 701 Your Apps and Evolving Network Security Standards.

> It can be resolved by adding [`NSAllowsArbitraryLoads`]

Some sort of ATS exception would have been required even on earlier systems.

> For testing purpose, we use a self-signed cert …

I strongly recommend against using a self-signed certificate for testing. There is another, better approach you can use, namely, a custom CA. QA1948 HTTPS and Test Servers discusses the problems with using a self-signed certificate, explains why using a custom CA is better, and gives links to resources that'll help you set this up.
Share and Enjoy
—
Quinn "The Eskimo!"
Apple Developer Relations, Developer Technical Support, Core OS/Hardware

```
let myEmail = "eskimo" + "1" + "@apple.com"
```

Actions ▾ This helped me (0)

Level 1
(0 points)

Dimitrij
Sep 27, 2017 5:24 AM
(in response to eskimo)

Hi eskimo,

we use self signed certificate too with tls 1.2 and sha256.
We have ca. 2600 Devices in our enveronment. how can we trust our certificate with on 2500 devices ? we use airwatch and ios 11 dont trust our root certificate. ios 10 works fine.

Actions This helped me (0)

eskimo
Sep 28, 2017 1:33 AM
(in response to Dimitrij)

Apple Staff
(13,265 points)

> how can we trust our certificate with on 2500 devices ?

This is a very different question from the one posed by hanxuanfromtoronto and it's somewhat outside of my area of expertise.
Most folks with that many devices use some sort of MDM-based management solution to manage them. That management solution will let deploy a configuration profile containing your root certificate to those devices.
It sounds like you're trying to do that and it's not working. If that's the case then you'll need to get in touch with folks experienced in this sort of thing:

- The support channel for the management solution you're using
- AppleCare's enterprise support options (there's some links in this post)
- The Apple Support Communities, run by AppleCare, and specifically the *in Business and Education* topic areas

Share and Enjoy
—
Quinn "The Eskimo!"
Apple Developer Relations, Developer Technical Support, Core OS/Hardware
`let myEmail = "eskimo" + "1" + "@apple.com"`

Actions This helped me (0)

Dimitrij P.
Nov 2, 2017 4:28 AM
(in response to eskimo)

Level 1
(0 points)

in ios 11.2 resolved our issue. under ios 11.1 we receive a message in safari " this connection is not private" in ios 11.2 we didnt. what happen in ios 11.2 ? In Relase Notes i found nothing.

Actions This helped me (0)

christopher_james
Oct 4, 2017 7:00 AM
(in response to eskimo)

Level 1
(0 points)

Hi,

I have a problem very similar to the one hanxuanfromtoronto has. When starting a network connection using URLSession on iOS11 with a self-signed certificate, I receive the _kCFStreamErrorCodeKey=-9802 while it used to work fine on iOS10. I verify the server certificate this way:

```
    func urlSession(_ session: URLSession,
                    didReceive challenge: URLAuthenticationChallenge
                    completionHandler: @escaping (URLSession.AuthCha
    {
        if challenge.protectionSpace.authenticationMethod == NSURLAu
            if challenge.protectionSpace.host == myserver {

                if let certFile = Bundle.main.path(forResource: "my-
                    let data = try? Data(contentsOf: URL(fileURLWith
                    let cert = SecCertificateCreateWithData(nil, dat
                    let trust = challenge.protectionSpace.serverTrus
                {
                    SecTrustSetAnchorCertificates(trust, [cert] as C
                    completionHandler(.useCredential, URLCredential(
                } else {
                    completionHandler(.cancelAuthenticationChallenge
                }
            } else {
                completionHandler(.performDefaultHandling, nil)
            }
        }
    }
```

```
    func urlSession(_ session: URLSession,
                    didReceive challenge: URLAuthenticationChallenge
                    completionHandler: @escaping (URLSession.AuthCha
    {
        if challenge.protectionSpace.authenticationMethod == NSURLAu
            if challenge.protectionSpace.host == myserver {

                if let certFile = Bundle.main.path(forResource: "my-
                    let data = try? Data(contentsOf: URL(fileURLWith
                    let cert = SecCertificateCreateWithData(nil, dat
                    let trust = challenge.protectionSpace.serverTrus
                {
                    SecTrustSetAnchorCertificates(trust, [cert] as C
                    completionHandler(.useCredential, URLCredential(
                } else {
                    completionHandler(.cancelAuthenticationChallenge
                }
            } else {
                completionHandler(.performDefaultHandling, nil)
            }
        }
    }
```

Some investigation showed that setting the following in my Info.plist circumvents the problem:

```
    <key>NSAppTransportSecurity</key>
    <dict>
         <key>NSAllowsArbitraryLoads</key>
         <false/>
         <key>NSExceptionDomains</key>
         <dict>
              <key>my-server.com</key>
              <dict>
                   <key>NSExceptionRequiresForwardSecrecy</key>
                   <false/>
              </dict>
         </dict>
    </dict>
```

```
    <key>NSAppTransportSecurity</key>
    <dict>
         <key>NSAllowsArbitraryLoads</key>
         <false/>
         <key>NSExceptionDomains</key>
         <dict>
              <key>my-server.com</key>
              <dict>
                   <key>NSExceptionRequiresForwardSecrecy</key>
                   <false/>
              </dict>
         </dict>
    </dict>
```

This suggests that the cipher suite my server uses seems to be deprecated in iOS11. So I tried the following:

```
$ curl https://my-server.com/some/path -k -v
...
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
* Server certificate: my-server.com
...
```

```
$ curl https://my-server.com/some/path -k -v
...
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
* Server certificate: my-server.com
...
```

So, my server actually supports the cipher suite TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, which is said to be supported here: https://developer.apple.com/library/content/documentation/General/Reference/InfoPlistK
SW57

I also checked my server certificate, it is using SHA-256 and a key length of 2048 bit so this should be fine (additionally, if the certificate was weak, setting NSExceptionRequiresForwardSecrecy to false should not solve the problem?).

Watching the WWDC 2017 Session 701 mentioned by eskimo did not get me further either.

Maybe the documentation I linked to is not updated for iOS 11 and the cipher suite my server uses is actually deprecated now? Or am I just getting this wrong?

Thanks for help and suggestions!

Actions

This helped me (0)

eskimo

Oct 5, 2017 12:45 AM

(in response to christopher_james)

Apple Staff

(13,265 points)

> Some investigation showed that setting the following in my Info.plist circumvents the problem:

Well, *that's* weird. Setting `NSExceptionRequiresForwardSecrecy` shouldn't magically allow you to connect to a server whose root is not trusted. Doing that requires either `NSExceptionAllowsInsecureHTTPLoads` or one of the global 'arbitrary loads' settings.

> Maybe the documentation I linked to is not updated for iOS 11 and the cipher suite my server uses is actually deprecated now?

No, `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256` is a fine cypher suite.

If the server in question is available on the wider Internet and you want to publicise that fact, please post its address. Otherwise you should open a DTS tech support incident and I can look help you out privately.

Share and Enjoy

—

Quinn "The Eskimo!"

Apple Developer Relations, Developer Technical Support, Core OS/Hardware

`let myEmail = "eskimo" + "1" + "@apple.com"`

Actions

This helped me (0)

cjoseph

Oct 5, 2017 10:27 AM

(in response to eskimo)

Level 1

(0 points)

Thanks eskimo & christopher_james — I'm working on the same project as hanxuanfromtoronto and the NSExceptionRequiresForwardSecrecy did in fact resolve our issue. I found this document on TLS changes in iOS 11 and it sounds like the issue is our server doesn't support the forward secrecy stuff. I'm going to have our server-side folks look into that but at least we've resolved it for the meantime!

Actions This helped me (0)

livestar
Nov 7, 2017 12:31 AM
(in response to cjoseph)

Level 1
(0 points)

I have a question to you.

You said you resolved TLS communication problems in Oct 5.
Is this problem solved in the release up to iOS version 11.1?
When it is not reflected in the release, when will it be reflected?

Some customers said they cannot login to our application or communication becomes unstable after updating iOS to version 11(maybe their OS version is up to 11.0.2).
I think this problem is related to TLS communication problems as you resolved, so I want to confirm this problem is already resolved and reflected to latest OS version or not.

Actions This helped me (0)

christopher_james
Oct 7, 2017 2:48 PM
(in response to eskimo)

Level 1
(0 points)

I did some more investigation and found out the problem is, at least in my case, acutally about the certificate and not the cipher suite. I changed my server to use a system-trusted CA-signed certificate, toggled off the custom root-certificate verification code inside the app and now it works.

The strange thing about this is that NSExceptionRequiresForwardSecrecy solved the issue, since as far as I know this is all about the used cipher suite and not about the certificate. Also, I double checked that my certificate fulfills the ATS requirements (no SHA-1, at least 2048bit), so in theory it should be possible to connect. The error _kCFStreamErrorCodeKey=-9802 (errSSLFatalAlert) is also not quite helpful.

Is there some documentation on what NSExceptionRequiresForwardSecrecy actually does? Seems like it is slightly more than turning off some old cipher suites. And is there any list about when exactly errSSLFatalAlert (-9802) is thrown?

Thanks in advance!

Actions

This helped me (0)

Apple Staff
(13,265 points)

eskimo

Oct 9, 2017 1:23 AM

(in response to christopher_james)

> I changed my server to use a system-trusted CA-signed certificate, toggled off the custom root-certificate verification code inside the app and now it works.

To be clear, this is what I recommend in virtually all cases. Disabling the default HTTPS server trust evaluation and relying on your own is very error prone, and the consequences of such an error can be very bad.

> Is there some documentation on what `NSExceptionRequiresForwardSecrecy`actually does?

That's a tricky question to answer. The *documented* effect of `NSExceptionRequiresForwardSecrecy` is that it enables some extra cypher suites. As you've noticed, it can have other effects (we saw this back when ATS was introduced; see this thread for details). It's best not to rely on those.

> Seems like it is slightly more than turning off some old cipher suites. And is there any list about when exactly `errSSLFatalAlert`(-9802) is thrown?

`errSSLFatalAlert` means exactly what it says on the tin: the remote peer disconnected the connection with a fatal alert. Alas, this doesn't help you much when it comes to debugging because other factors can cause that. For example, I've seen situations where servers throw a fatal alert because the client had the temerity to request a higher TLS version number.
Which isn't to say that iOS is behaving correctly here, just that you can't learn much from the error.
If you want to dig into this you'll need to look at:

- A system log, to see if the iOS TLS code has logged any info about what went wrong
- A packet trace, to compare the failing and non-failing cases and to see if there really was a fatal alert and, if so, what the alert code was

Share and Enjoy

—

Quinn "The Eskimo!"

Apple Developer Relations, Developer Technical Support, Core OS/Hardware

```
let myEmail = "eskimo" + "1" + "@apple.com"
```

Actions

This helped me (0)

PeterStegnar

Oct 16, 2017 12:55 AM

(in response to eskimo)

Level 1
(0 points)

If we go back to the main topic how to enable in iOS 11 to work with self signed certificate.

I tried with

```
NSExceptionAllowsInsecureHTTPLoads
NSExceptionAllowsInsecureHTTPLoads
```

setting, but with no success.

I have found out that ATS exception domains wont work with IP, so I need actual domain. I have used test "bad ssl" site: https://self-signed.badssl.com/. I am using this site because I want to be sure everything is fine from server perspective. So I have configure:

```
<key>NSAppTransportSecurity</key>
<dict>
  <key>NSAllowsLocalNetworking</key>
  <true/>
  <key>NSExceptionDomains</key>
  <dict>
    <key>badssl.com</key>
    <dict>
      <key>NSIncludesSubdomains</key>
      <true/>
      <key>NSExceptionRequiresForwardSecrecy</key>
      <false/>
      <key>NSExceptionAllowsInsecureHTTPLoads</key>
      <true/>
    </dict>
  </dict>
  <key>NSAllowsArbitraryLoadsInWebContent</key>
  <true/>
  <key>NSAllowsArbitraryLoads</key>
  <true/>
```

```
</dict>
```

The problem is that this still does not work. How can I make it work? With what setting? I am using WKWebView with Cordova.

Actions

This helped me (0)

Apple Staff
(13,265 points)

eskimo
Oct 17, 2017 1:47 AM
(in response to PeterStegnar)

> If we go back to the main topic how to enable in iOS 11 to work with self signed certificate.

Sure.

> I have found out that ATS exception domains wont work with IP, so I need actual domain.

Correct.

> I have used test "bad ssl" site .... I am using this site because I want to be sure everything is fine from server perspective.

Yep, that's an excellent choice.
Here's what I did:

1. I created a new app with the following ATS dictionary:

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSExceptionDomains</key>
    <dict>
        <key>self-signed.badssl.com</key>
        <dict>
            <key>NSExceptionAllowsInsecureHTTPLoads
            <true/>
        </dict>
    </dict>
</dict>
```

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSExceptionDomains</key>
    <dict>
        <key>self-signed.badssl.com</key>
        <dict>
            <key>NSExceptionAllowsInsecureHTTPLoads
            <true/>
        </dict>
    </dict>
</dict>
```

-

2. I set the view controller to this:

```
<span>import UIKit


class MainViewController : UITableViewController, U


var _session: URLSession? = nil
var session: URLSession {
    if let session = self._session {
        return session
    }
    let config = URLSessionConfiguration.default
    config.requestCachePolicy = .reloadIgnoringLoca
    let session = URLSession(configuration: config,
    self._session = session
    return session
}


override func tableView(_ tableView: UITableView, d
    let url = URL(string: "</span><a class="jive-li
    let request = URLRequest(url: url)
    self.session.dataTask(with: request) { (_, resp
        if let error = error as NSError? {
            NSLog("transport error, error: %@ / %d"
            return
        }
        let response = response as! HTTPURLResponse
        NSLog("response, statusCode: %d", response.
    }.resume()
    self.tableView.deselectRow(at: indexPath, anima
}


func urlSession(_ session: URLSession, didReceive c
    let method = challenge.protectionSpace.authenti
    let host = challenge.protectionSpace.host
    NSLog("challenge %@ for %@", method, host)
    switch (method, host) {
    case (NSURLAuthenticationMethodServerTrust, "se
        let trust = challenge.protectionSpace.serve
        let credential = URLCredential(trust: trust
        completionHandler(.useCredential, credentia
    default:
        completionHandler(.performDefaultHandling,
    }
}

}
</span>
```

```
<span>import UIKit


class MainViewController : UITableViewController, U


var _session: URLSession? = nil
var session: URLSession {
    if let session = self._session {
        return session
    }
    let config = URLSessionConfiguration.default
    config.requestCachePolicy = .reloadIgnoringLoca
    let session = URLSession(configuration: config,
    self._session = session
    return session
}


override func tableView(_ tableView: UITableView, d
    let url = URL(string: "</span><a class="jive-li
    let request = URLRequest(url: url)
    self.session.dataTask(with: request) { (_, resp
        if let error = error as NSError? {
            NSLog("transport error, error: %@ / %d"
            return
        }
        let response = response as! HTTPURLResponse
        NSLog("response, statusCode: %d", response.
    }.resume()
    self.tableView.deselectRow(at: indexPath, anima
}


func urlSession(_ session: URLSession, didReceive c
    let method = challenge.protectionSpace.authenti
    let host = challenge.protectionSpace.host
    NSLog("challenge %@ for %@", method, host)
    switch (method, host) {
    case (NSURLAuthenticationMethodServerTrust, "se
        let trust = challenge.protectionSpace.serve
        let credential = URLCredential(trust: trust
        completionHandler(.useCredential, credentia
    default:
        completionHandler(.performDefaultHandling,
    }
}

}
</span>
```

-

3. I ran the app on iOS 11 and started the test.

4. It printed:

```
2017-10-17 09:34:10.986113+0100 BadSSL[59827:135968
2017-10-17 09:34:11.264039+0100 BadSSL[59827:135968
```

```
2017-10-17 09:34:10.986113+0100 BadSSL[59827:135968
2017-10-17 09:34:11.264039+0100 BadSSL[59827:135968
```

-

5. I then took that authentication challenge handling code and plugged it into my standard WKWebView test app.

```
func webView(_ webView: WKWebView, didReceive chall
    let method = challenge.protectionSpace.authenti
    let host = challenge.protectionSpace.host
    NSLog("challenge %@ for %@", method, host)
    switch (method, host) {
    case (NSURLAuthenticationMethodServerTrust, "se
        let trust = challenge.protectionSpace.serve
        let credential = URLCredential(trust: trust
            completionHandler(.useCredential, credentia
        default:
            completionHandler(.performDefaultHandling,
        }
    }
```

```
    func webView(_ webView: WKWebView, didReceive chall
        let method = challenge.protectionSpace.authenti
        let host = challenge.protectionSpace.host
        NSLog("challenge %@ for %@", method, host)
        switch (method, host) {
        case (NSURLAuthenticationMethodServerTrust, "se
            let trust = challenge.protectionSpace.serve
            let credential = URLCredential(trust: trust
            completionHandler(.useCredential, credentia
        default:
            completionHandler(.performDefaultHandling,
        }
    }
```

-

6. I set that app's ATS exceptions as per step 1.

7. I ran the app, navigated to the site, and saw the big red `self-signed.badssl.com` page.

As far as I can tell this all seems to be working just fine.

WARNING The above *completely disables* security checking on the `self-signed.badssl.com` domain. That's very poor form in a production app, but it makes sense for this test because I'm trying to show that you can talk to an HTTPS server regardless of its certificate. In a production app you'd want to narrow this exception by explicitly testing the certificate of the server. Or give your server a proper, CA-issued certificate, which avoids all of this drama.

Share and Enjoy

—

Quinn "The Eskimo!"

Apple Developer Relations, Developer Technical Support, Core OS/Hardware

`let myEmail = "eskimo" + "1" + "@apple.com"`

Actions

This helped me (0)

Level 1

(0 points)

devinWang

May 24, 2018 10:28 AM

(in response to eskimo)

Hi eskimo,

my api is https://loneworker.proraeguardian.com:9093, when set NSExceptionRequiresForwardSecrecy to false, it's ok.

But our certificate is support PFS. i receive log blow when don't set ForwardSecrecy to false.

```
2018-05-25 00:57:16.653797+0800 SC[4437:3189843] ATS failed system trust
2018-05-25 00:57:16.653927+0800 SC[4437:3189843] System Trust failed for
[3:0x1c016ee80]
2018-05-25 00:57:16.654134+0800 SC[4437:3189843] TIC SSL Trust Error
[3:0x1c016ee80]: 3:0
2018-05-25 00:57:16.654486+0800 SC[4437:3189843] NSURLSession/NSURLConnection
HTTP load failed (kCFStreamErrorDomainSSL, -9802)
2018-05-25 00:57:16.654597+0800 SC[4437:3189843] Task <358A4316-A434-420D-
BD92-FFBCE3391101>.<0> HTTP load failed (error code: -1200 [3:-9802])
2018-05-25 00:57:16.654952+0800 SC[4437:3189845] NSURLConnection finished with
error - code -1200
```

Do you have any ideas?

Actions

This helped me (0)

Piaojin

Jun 9, 2019 11:42 PM

(in response to eskimo)

Level 1
(0 points)

Hi, Eskimo. Recently, My app will enable ATS and support ATS. I use `/usr/bin/nscurl --ats-diagnostics my_url` to verify server API. I get the following report and the question is how can I know the server API is conformed to ATS or not?

```
Starting ATS Diagnostics

Configuring ATS Info.plist keys and displaying the result of HTTPS loads to xxxxx.
A test will "PASS" if URLSession:task:didCompleteWithError: returns a nil error.
Use '--verbose' to view the ATS dictionaries used and to display the error received in
URLSession:task:didCompleteWithError:.
================================================================================

Default ATS Secure Connection
---
ATS Default Connection
Result : PASS
---

================================================================================

Allowing Arbitrary Loads
```

```
---
Allow All Loads
Result : PASS
---

==============================================================

Configuring TLS exceptions for xxxx

---
TLSv1.3
2019-06-10 14:05:15.205 nscurl[96230:2204640] NSURLSession/NSURLConnection HTTP
load failed (kCFStreamErrorDomainSSL, -9858)
Result : FAIL
---

---
TLSv1.2
Result : PASS
---

---
TLSv1.1
Result : PASS
---

---
TLSv1.0
Result : PASS
---

==============================================================

Configuring PFS exceptions for xxx

---
Disabling Perfect Forward Secrecy
Result : PASS
---

==============================================================

Configuring PFS exceptions and allowing insecure HTTP for xxxx

---
Disabling Perfect Forward Secrecy and Allowing Insecure HTTP
Result : PASS
```

```
---

================================================================================

Configuring TLS exceptions with PFS disabled for xxx

---
TLSv1.3 with PFS disabled
2019-06-10 14:05:27.286 nscurl[96230:2204640] NSURLSession/NSURLConnection HTTP
load failed (kCFStreamErrorDomainSSL, -9858)
Result : FAIL
---

---
TLSv1.2 with PFS disabled
Result : PASS
---

---
TLSv1.1 with PFS disabled
Result : PASS
---

---
TLSv1.0 with PFS disabled
Result : PASS
---

================================================================================

Configuring TLS exceptions with PFS disabled and insecure HTTP allowed for xxx

---
TLSv1.3 with PFS disabled and insecure HTTP allowed
2019-06-10 14:05:43.052 nscurl[96230:2204700] NSURLSession/NSURLConnection HTTP
load failed (kCFStreamErrorDomainSSL, -9858)
Result : FAIL
---

---
TLSv1.2 with PFS disabled and insecure HTTP allowed
Result : PASS
---

---
TLSv1.1 with PFS disabled and insecure HTTP allowed
Result : PASS
---
```

TLSv1.0 with PFS disabled and insecure HTTP allowed
Result : PASS

===

Thank you.

Actions

This helped me (0)

NTSMK
Oct 31, 2017 11:38 PM
(in response to hanxuanfromtoronto)

Level 1
(0 points)

I use AFNetworking, with you the same problem, how can I solve it?

Actions

This helped me (0)

eskimo
Nov 1, 2017 1:15 AM
(in response to NTSMK)

Apple Staff
(13,265 points)

I've got a couple of suggestions for you here:

- You can ask for help from the support channel for AFNetworking.
- You can dig into the AFNetworking code to work out how it interacts with the operation system, thus allowing you to map the steps I've described (in my 17 Oct post) into the AFNetworking world.

Share and Enjoy
—
Quinn "The Eskimo!"
Apple Developer Relations, Developer Technical Support, Core OS/Hardware
`let myEmail = "eskimo" + "1" + "@apple.com"`

Actions

This helped me (0)

Level 1
(0 points)

JimCarry361

Feb 26, 2019 3:17 AM

(in response to hanxuanfromtoronto)

Everything above doesn't work, if you work with AVFoundation API. AVPlayer in particular. I've added set 'allow arbitary loads' to yes, added exception domains(to be sure it works). Nothing helps, got an error "Error Domain=NSOSStatusErrorDomain Code=-1202 "(null)"}, NSLocalizedDescription=The certificate for this server is invalid." Other requests via https to the same server works, while AVPlayer API - doesn't. Any idea why?

Actions

This helped me (0)

Go to original post

More Like This

- make an application which will communicate seamlessly over HTTPS with multiple servers.
- Certificate Transparency
- Re: Loading .onion on device "The Internet connection appears to be offline"
- Error Domain=NSURLErrorDomain Code=-1200 "An SSL error has occurred and a secure connection to the server cannot be made."
- Re: Ask user to Install and trust Self Signed Certificate from Private CA