<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>MartinBosslet (Martin Bosslet)</td>
</tr>
<tr>
<td>Target version:</td>
<td>2.0.0</td>
</tr>
</tbody>
</table>

**Backport:**

#### Description

**not_after can not later than 2038-01-19 11:14:07**

**verify can pass wrong certificate**

require 'OpenSSL'

class Rsa
  attr_reader :key, :cert
  @@sha = OpenSSL::Digest::SHA1.new
  @@aes = OpenSSL::Cipher.new("aes-128-ofb")

def initialize serial, issuer=nil
  @key = OpenSSL::PKey::RSA.new(1024)
  @cert = OpenSSL::X509::Certificate.new
  @cert.version = 2 # RFC 5280 - v3
  @cert.serial = serial
  @cert.subject = OpenSSL::X509::Name.parse "CN=#{serial}"
  @cert.issuer = issuer.nil? ? @cert.subject : issuer
  @cert.public_key = @key.public_key
  @cert.not_before = Time.now
  @cert.not_after = Time.mktime(2038, 1, 19, 11, 14, 7) # second = 8 ==> Fail!
  @cert.sign(@key, @@sha) if issuer.nil
end

def sign key
  @cert.sign(key, @@sha)
end

def sign_encrypt(plain, obj_cert)
  signed = OpenSSL::PKCS7::sign(@cert, @key, plain)
  encrypted = OpenSSL::PKCS7::encrypt([obj_cert], signed.to_s, @@aes)
end

def decrypt_verify(received, obj_cert, ca_cert)
  encrypted = OpenSSL::PKCS7.new(received)
  decrypted = encrypted.decrypt(@key, @cert)
  signed = OpenSSL::PKCS7.new(decrypted)
  cert_store = OpenSSL::X509::Store.new.add_cert(ca_cert)
  plain = signed.data if signed.verify([obj_cert], cert_store)
end

c = Rsa.new(1)
a = Rsa.new(11, c.cert.issuer)
a.sign ca.key
right = Rsa.new(12, c.cert.issuer)
right.sign ca.key
fa = Rsa.new(3)
wrong = Rsa.new(33, fa.cert.issuer)
```ruby
wrong.sign fa.key  # Don't sign indeed!

plain = "Something's wrong."
signed_encrypted = right.sign_encrypt(plain, alice.cert)
recovered = alice.decrypt_verify(signed_encrypted, wrong.cert, ca.cert)  # wrong should be right
puts recovered==plain ? "It's okay!" : recovered;

Related issues:
Related to Ruby master - Bug #6571: Time.mktime Y2K38 problem on 1.9.3p125 i3...  Closed

History
#1 - 03/08/2012 02:34 PM - mghomn (Justin Peal)

not_after can not later than 2038-01-19 11:14:07

require 'OpenSSL'
class Rsa
  attr_reader :key, :cert
  @@sha = OpenSSL::Digest::SHA1.new
  @@aes = OpenSSL::Cipher.new("aes-128-ofb")

  def initialize serial, issuer=nil
    @key = OpenSSL::PKey::RSA.new(1024)
    @cert = OpenSSL::X509::Certificate.new
    @cert.version = 2  # RFC 5280 - v3
    @cert.serial = serial
    @cert.subject = OpenSSL::X509::Name.parse "CN=#{serial}"
    @cert.issuer = issuer==nil ? @cert.subject : issuer
    @cert.public_key = @key.public_key
    @cert.not_before = Time.now
    @cert.not_after = Time.mktime(2038, 1, 19, 11, 14, 7)  # second = 8 ==> Fail!
    @cert.sign(@key, @sha) if issuer==nil
  end

  def sign key
    @cert.sign(key, @@sha)
  end

  def sign_encrypt(plain, obj_cert)
    signed = OpenSSL::PKCS7::sign(@cert, @key, plain)
    encrypted = OpenSSL::PKCS7::encrypt([obj_cert], signed.to_s, @@aes)
  end

  def decrypt_verify(received, obj_cert, ca_cert)
    encrypted = OpenSSL::PKCS7.new(received)
    decrypted = encrypted.decrypt(@key, @cert)
    signed = OpenSSL::PKCS7.new(decrypted)
    cert_store = OpenSSL::X509::Store.new.add_cert(ca_cert)
    plain = signed.data if signed.verify([obj_cert], cert_store, nil, OpenSSL::PKCS7::NOINTERN | OpenSSL::PKCS7::NOCHAIN)
  end

end

ca = Rsa.new(1)
alice = Rsa.new(11, ca.cert.issuer)
alice.sign ca.key
right = Rsa.new(12, ca.cert.issuer)
right.sign ca.key
fa = Rsa.new(3)
wrong = Rsa.new(33, fa.cert.issuer)
wrong.sign fa.key

plain = "Something's wrong."
signed_encrypted = right.sign_encrypt(plain, alice.cert)
recovered = alice.decrypt_verify(signed_encrypted, right.cert, ca.cert)
puts recovered==plain ? "It's okay!" : recovered;
recovered = alice.decrypt_verify(signed_encrypted, wrong.cert, ca.cert)  # wrong should be right
puts recovered==plain ? "It's okay!" : recovered;

#2 - 03/08/2012 06:49 PM - MartinBosslet (Martin Bosslet)
```
Hi Justin,

The behavior you encountered is not an error. When you sign the PKCS7, the signing certificate will be included in the resulting SignedData structure. You can see that:

```ruby
def decrypt_verify(received, obj_cert, ca_cert)
  encrypted = OpenSSL::PKCS7.new(received)
  decrypted = encrypted.decrypt(@key, @cert)
  signed = OpenSSL::PKCS7.new(decrypted)
  cert_store = OpenSSL::X509::Store.new.add_cert(ca_cert)
  signed.certificates.each { |c| p c } # => the signing certificate is in there
  plain = signed.data if signed.verify([obj_cert], cert_store)
end
```

When the PKCS7 is verified later on, OpenSSL will at first look through the certificates you provided and then look in the SignedData itself if it can find the signing certificate there. It does, so it ignores your additional certificate. With the signing certificate included,

```ruby
signed.verify(nil, cert_store)
```

will also succeed, and this is expected. If you want it to behave differently, you may either use the flags as in your second example, or you might sign the data without including the signing certificates.

Regarding the time issue, you ran into the Y2K38 problem there. This shouldn't be a problem anymore with your Ruby version, and it works on my Linux machine, could be that it is a problem specific to Windows. I'll close this issue and open a separate one for the time problem as they are not related.

#5 - 06/11/2012 01:21 AM - MartinBosslet (Martin Bosslet)

#6 - 12/06/2012 05:17 PM - mghomn (Justin Peal)

Thanks a lot!