Defeating OCSP With The Character '3'

Moxie Marlinspike

07/29/09

Abstract

This paper presents some simple tricks for defeating Online Certificate Status Protocol (OCSP) checks against forged or revoked certificates.

Some Background

The OCSP protocol was developed to better support the needs of certificate revocation in a world where more certificates are being issued by a greater diversity of Certificate Authorities. As more and more certificates need to be revoked, the old methods of maintaining large certificate revocation lists that need to be manually updated for each Certificate Authority are no longer effective.

The OCSP protocol was designed to support light-weight requests from clients who are being presented with certificates, in order to check the validity of those certificates in real-time. A web-browser such as Firefox, for instance, when presented with a certificate for www.ebay.com that it hasn't seen recently, will – before accepting the certificate – make a quick connection to the certificate's issuer to ask if that certificate should still be considered valid.

If we have forged or otherwise surreptitiously obtained a certificate through fraudulent means, this protocol can be deadly if those certificates are discovered and revoked by their issuers.

Some Useful Tricks

At a glance, the basic OCSP response structure is as follows:

```
OCSPResponse ::= SEQUENCE {
responseStatus OCSPResponseStatus,
responseBytes [0] EXPLICIT ResponseBytes OPTIONAL
}
```

If we look at the innermost ResponseBytes structure:

```
BasicOCSPResponse ::= SEQUENCE {
tbsResponseData ResponseData,
signatureAlgorithm AlgorithmIdentifier,
```

signature BIT STRING,
certs [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL
}

We see that within the ResponseBytes structure, there is a signature by the Certificate Authority which is being queried. The interesting thing to note, however, is that this signature only covers the fields in ResponseData (which is optional). It does not cover the fields in the ResponseStatus structure.

If we look at ResponseStatus, we see that there are a number of possible status options:

OCSPResponseStatus ::= ENUMERATED {
successful (0), –Response has valid confirmations
malformedRequest (1), –Illegal confirmation request
internalError (2), –Internal error in issuer
tryLater (3), –Try again later
sigRequired (5), –Must sign the request
unauthorized (6) –Request unauthorized
}

If we wish to forge an OCSP response, we might have trouble choosing to respond with "successful," since that would require that we include a subsequent ResponseData section which contains a signature we might not be able to generate. The status values "malformedRequest," "internalError," "sigRequired," and "unauthorized" do not require that we include a subsequent ResponseData section, but they all seem to convey error conditions, and sound kind of bad. But the value "tryLater" doesn't sound bad at all, or at least doesn't seem to convey any sense of a problem with the certificate that's been queried, and it does not require that we include a ResponseData field either.

As an attacker, it is thus possible for us to intercept any OCSP request and send a "tryLater" response without having to generate a signature of any kind. The composition of the response is literally just the the OCSPResponseStatus for the "tryLater" condition, which is simply the single-byte ASCII character '3'.

Most OCSP implementations will accept this response without generating any kind of warning to the user that something might be amiss, thus defeating the protocol.

Exploitation

The SSL/TLS man-in-the-middle attack tool, sslsniff[1], has been updated to support this mode of attack against OCSP. If running in "targeted certificate mode," it will intercept and forge "tryLater" responses for any OCSP request to the certificates that are being deployed in a man-in-the-middle attack.

[1] http://www.thoughtcrime.org/software/sslsniff/