Multiple OCSP Responses

In TLS Handshake

draft-pettersen-tls-ext-multiple-ocsp-03

Yngve N. Pettersen

Opera Software ASA
Problem statement

- Using TLS OCSP Stapling for intermediate CA certificates would give more timely revocation information to clients
- Will improve user experience, lower workload for CAs, slightly increase bandwidth usage for sites
- The current status_request extension only allows OCSP for the site certificate, and adding more methods is not feasible due to design limitations
- A new extension that allows multiple status methods is needed
Current status of -03

- Implementation revealed a design problem with future addition of new status request methods
- Older implementations would not be able to read the new status request records
- Fixed this problem by adding a length field for the request field in status request records
- The same design problem was found to affect two existing extensions: SNI and Trusted CA
Way forward

- The document is based on an existing extension
- The document is near technical completion
- Request that draft be handled as TLS WG item since it replaces an existing RFC 6066 extension
Appendix: The variant record issue

Client

enum{
    Foo, Bar, Wha1;
} Typ

struct {
    Typ rec_typ;
    Select(rec_typ)
    {
        case Foo: opaque food<1..2^8-1>;
        case Bar: opaque barge<1..2^16-1>;
        case Wha1:
            opaque whatsths<1..2^8-1>;
            opaque whatstht<1..2^16-1>;
    } payload;
}Rec;

Rec Recs<1..2^16-1>;

Server

enum{
    Foo, Bar;
} Typ

struct {
    Typ rec_typ;
    Select(rec_typ)
    {
        case Foo: opaque food<1..2^8-1>;
        case Bar: opaque barge<1..2^16-1>;
    } payload;
}Rec;

Rec Recs<1..2^16-1>;
## Appendix: The variant record issue

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>vector length</td>
<td>OK</td>
</tr>
<tr>
<td>Bar</td>
<td>OK</td>
</tr>
<tr>
<td>Payload.barge</td>
<td>OK</td>
</tr>
<tr>
<td>ABCDEF</td>
<td>Payload.barge ABCDEF</td>
</tr>
<tr>
<td>Wha1</td>
<td>??? Unknown type. Will ignore following content.</td>
</tr>
<tr>
<td>Payload.whatsths</td>
<td>Huh? What's this? Don't know how to parse it.</td>
</tr>
<tr>
<td>HJKLMN</td>
<td>Exception!</td>
</tr>
<tr>
<td>Payload.whatstht</td>
<td>Exception!</td>
</tr>
<tr>
<td>OPQRS</td>
<td>Exception!</td>
</tr>
<tr>
<td>Foo</td>
<td>Exception!</td>
</tr>
<tr>
<td>Payload.food</td>
<td>Exception!</td>
</tr>
<tr>
<td>TUVWXYZ</td>
<td>Exception!</td>
</tr>
</tbody>
</table>
Appendix: The variant record issue

```c
enum{
    Foo, Bar, Wha1;
} Typ

struct {
    Typ rec_typ;
    uint16 payload_length;
    Select(rec_typ) {
        case Foo: opaque food<1..2^8-1>;
        case Bar: opaque barge<1..2^16-1>;
        case Wha1:
            opaque whatsths<1..2^8-1>;
            opaque whatstht<1..2^16-1>;
    }
    payload;
} Rec;

Rec Recs<1..2^16-1>;
```

```c
enum{
    Foo, Bar;
} Typ

struct {
    Typ rec_typ;
    uint16 payload_length;
    Select(rec_typ) {
        case Foo: opaque food<1..2^8-1>;
        case Bar: opaque barge<1..2^16-1>;
        }
    payload;
} Rec;

Rec Recs<1..2^16-1>;
```
# Appendix: The variant record issue

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>vector length</td>
<td>OK</td>
</tr>
<tr>
<td>Bar</td>
<td>OK</td>
</tr>
<tr>
<td>length of Payload</td>
<td>OK</td>
</tr>
<tr>
<td>Payload.barge ABCDEF</td>
<td>OK</td>
</tr>
<tr>
<td>Payload.barge ABCDEF</td>
<td>OK</td>
</tr>
<tr>
<td>Wha1</td>
<td>??? Unknown type. Will ignore following content.</td>
</tr>
<tr>
<td>length of Payload</td>
<td>OK</td>
</tr>
<tr>
<td>Payload.whatsths HJKL</td>
<td>Ignoring this, since I don't know what it is</td>
</tr>
<tr>
<td>Payload.whatstht OPQR</td>
<td>Ignoring this, since I don't know what it is</td>
</tr>
<tr>
<td>Foo</td>
<td>OK</td>
</tr>
<tr>
<td>length of Payload</td>
<td>OK</td>
</tr>
<tr>
<td>Payload.food TUVWXYZ</td>
<td>OK</td>
</tr>
<tr>
<td>Payload.food TUVWXYZ</td>
<td>OK</td>
</tr>
</tbody>
</table>