HW7

Benjamin Cape

August 2, 2022

Q1

Using Chrome...

Using Chrome we see that, after erasing the cache from the CRLs, we make a HTTP request to the OCSP server, which responds in time with the proper OCSP response: whether or not the certificate is valid.

Example from our Wireshark export:

```
Request:

554 10.973334   10.0.0.156  72.21.91.29 HTTP    417 GET
↪  /ME8wTTBLMEkwRzAHBgUrDgMCGgQUdLTnIxnHZZIVQER7x84%2BkMIYdusEFKSN5b58eeRwI20uKTStI1jc... HTTP/1.1

Response:

562 11.031332   72.21.91.29 10.0.0.156  OCSP    753 Response [Which encodes Revoked in the OCSP
↪  certStatus]
```

Using Firefox...

We also see an OCSP request being made.

```
197 1.069786    192.168.86.27   142.250.72.163  OCSP    499 Request

Frame 7380: 499 bytes on wire (3992 bits), 499 bytes captured (3992 bits) on interface en0, id 0
Ethernet II, Src: Apple_b5:bd:9e (a0:78:17:b5:bd:9e), Dst: Google_bd:88:de (60:b7:6e:bd:88:de)
Internet Protocol Version 4, Src: 192.168.86.27, Dst: 142.250.72.163
Transmission Control Protocol, Src Port: 59261, Dst Port: 80, Seq: 434, Ack: 703, Len: 433
Hypertext Transfer Protocol
    POST /gts1c3 HTTP/1.1\r\n
    Host: ocsp.pki.goog\r\n
    User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:103.0) Gecko/20100101 Firefox/103.0\r\n
    Accept: */*\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Content-Type: application/ocsp-request\r\n
    Content-Length: 84\r\n
    Connection: keep-alive\r\n
    Pragma: no-cache\r\n
    Cache-Control: no-cache\r\n
    \r\n
    [Full request URI: http://ocsp.pki.goog/gts1c3]
    [HTTP request 2/2]
    [Prev request in frame: 197]
    [Response in frame: 7390]
    File Data: 84 bytes
Online Certificate Status Protocol
    tbsRequest
        requestList: 1 item
            Request
```

```
            reqCert
                hashAlgorithm (SHA-1)
                    Algorithm Id: 1.3.14.3.2.26 (SHA-1)
                issuerNameHash: c72e798addff6134b3baed4742b8bbc6c0240763
                issuerKeyHash: 8a747faf85cdee95cd3d9cd0e24614f371351d27
                serialNumber: 0x00e284173e75d8dc9f0a1ec0fde6e03e8c
```

Following by the response:

```
Frame 7390: 768 bytes on wire (6144 bits), 768 bytes captured (6144 bits) on interface en0, id 0
Ethernet II, Src: Google_bd:88:de (60:b7:6e:bd:88:de), Dst: Apple_b5:bd:9e (a0:78:17:b5:bd:9e)
Internet Protocol Version 4, Src: 142.250.72.163, Dst: 192.168.86.27
Transmission Control Protocol, Src Port: 80, Dst Port: 59261, Seq: 703, Ack: 867, Len: 702
Hypertext Transfer Protocol
    HTTP/1.1 200 OK\r\n
    Content-Type: application/ocsp-response\r\n
    Date: Mon, 01 Aug 2022 16:30:20 GMT\r\n
    Cache-Control: public, max-age=14400\r\n
    Server: ocsp_responder\r\n
    Content-Length: 472\r\n
    X-XSS-Protection: 0\r\n
    X-Frame-Options: SAMEORIGIN\r\n
    \r\n
    [HTTP response 2/2]
    [Time since request: 0.142693000 seconds]
    [Prev request in frame: 197]
    [Prev response in frame: 290]
    [Request in frame: 7380]
    [Request URI: http://ocsp.pki.goog/gts1c3]
    File Data: 472 bytes
Online Certificate Status Protocol
    responseStatus: successful (0)
    responseBytes
        ResponseType Id: 1.3.6.1.5.5.7.48.1.1 (id-pkix-ocsp-basic)
        BasicOCSPResponse
            tbsResponseData
                responderID: byKey (2)
                producedAt: 2022-07-31 12:29:56 (UTC)
                responses: 1 item
                    SingleResponse
                        certID
                            hashAlgorithm (SHA-1)
                                Algorithm Id: 1.3.14.3.2.26 (SHA-1)
                            issuerNameHash: c72e798addff6134b3baed4742b8bbc6c0240763
                            issuerKeyHash: 8a747faf85cdee95cd3d9cd0e24614f371351d27
                            serialNumber: 0x00e284173e75d8dc9f0a1ec0fde6e03e8c
                        certStatus: good (0)
                        thisUpdate: 2022-07-31 12:29:55 (UTC)
                        nextUpdate: 2022-08-07 11:29:54 (UTC)
            signatureAlgorithm (sha256WithRSAEncryption)
                Algorithm Id: 1.2.840.113549.1.1.11 (sha256WithRSAEncryption)
            Padding: 0
            signature: 239a96a85b56543cd1486399ec19708fcaa7a641b5c985272e58a97b1452923989c00c06...
```

The Firefox OCSP seems to be more clear. It is very easy to replicate, even on a non-vm where there is lots of noise. Since normal calls do not usually make OCSP calls, starting wireshark before making a request in Firefox, and immediately stopping wireshark provides a relatively easy search in wireshark for the OCSP requests.

An interesting difference between Firefox and Chrome though: Chrome did not make any OCSP-protocol Requests. Rather, it was making OCSP requests through what appeared to be an HTTP abstraction layer.

Using Safari...

I couldn't get this working with safari, safari, even for a revoked website made an OCSP call that returned with the GOOD status.

Using openssl...

One can re-create curl, such that it always calls checks the CRLs for a given site (currently curl does not). This might lead to security concerns.

For example

```
# curl https://revoked.badssl.com/
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="shortcut icon" href="/icons/favicon-red.ico"/>
  <link rel="apple-touch-icon" href="/icons/icon-red.png"/>
  <title>revoked.badssl.com</title>
  <link rel="stylesheet" href="/style.css">
  <style>body { background: red; }</style>
</head>
<body>
<div id="content">
  <h1 style="font-size: 10vw;">
    revoked.<br>badssl.com
  </h1>
</div>

<div id="footer">
  The leaf certificate for this site has been revoked.
</div>

</body>
</html>
```

But this actually has a revoked certificate, which openssl does not check. ./scurl attempts to fix this.

```
#!/bin/bash

openssl s_client -servername $1 -connect $2:443 < /dev/null 2>&1 |  sed -n '/-----BEGIN/,/-----END/p' >
↪  ./certificate.pem

mkdir ./.scurl_tmp

openssl s_client -showcerts -servername $1 -connect $2:443 < /dev/null 2>&1 |  sed -n
↪  '/-----BEGIN/,/-----END/p' > ./.scurl_tmp/certs.txt

python3 ./build_chain.py ./.scurl_tmp/certs.txt ./chain.pem

OSCP_URL=$(openssl x509 -noout -ocsp_uri -in ./certificate.pem)

openssl ocsp -issuer chain.pem -cert certificate.pem -text -url $OSCP_URL
```

Try running:

```
./scurl.sh revoked.badssl.com www.revoked.badssl.com
```

Or any command where the first argument is the servername and the second argument is the url you are connecting to. We need the server, because in order to get all the information we need to allow for SNI (Server Name Indication).

We see using this new scurl tool:

```
./scurl.sh revoked.badssl.com www.revoked.badssl.com
mkdir: ./.scurl_tmp: File exists
```

```
OCSP Request Data:
    Version: 1 (0x0)
    Requestor List:
        Certificate ID:
          Hash Algorithm: sha1
          Issuer Name Hash: 74B4E72319C765921540447BC7CE3E90C21876EB
          Issuer Key Hash: A48DE5BE7C79E470236D2E2934AD2358DCF5317F
          Serial Number: 0D2E67A298853B9A5452E3A285A4572F
    Request Extensions:
        OCSP Nonce:
            04102C8A6AA80345CC57EA51AC4419EC2087
OCSP Response Data:
    OCSP Response Status: successful (0x0)
    Response Type: Basic OCSP Response
    Version: 1 (0x0)
    Responder Id: A48DE5BE7C79E470236D2E2934AD2358DCF5317F
    Produced At: Aug  1 01:24:58 2022 GMT
    Responses:
    Certificate ID:
      Hash Algorithm: sha1
      Issuer Name Hash: 74B4E72319C765921540447BC7CE3E90C21876EB
      Issuer Key Hash: A48DE5BE7C79E470236D2E2934AD2358DCF5317F
      Serial Number: 0D2E67A298853B9A5452E3A285A4572F
    Cert Status: revoked
    Revocation Time: Oct 27 21:38:48 2021 GMT
    This Update: Aug  1 01:09:01 2022 GMT
    Next Update: Aug  8 00:24:01 2022 GMT

    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        8d:4d:aa:ad:c5:5e:35:e1:a7:eb:67:11:c7:38:af:f7:d3:5f:
        88:59:4f:f4:f4:29:8c:0e:c3:e5:06:c2:39:f8:69:af:fd:17:
        43:b9:22:85:37:cd:7d:ee:dc:61:1e:59:7f:28:8b:d1:1e:dc:
        64:05:ff:6d:68:4f:03:f2:a2:e5:a3:fe:30:d6:c2:00:13:6a:
        24:98:db:ae:19:fc:d3:24:79:6b:52:b8:e3:03:01:8a:34:b5:
        19:77:85:c2:05:73:87:41:e0:2c:30:c7:ef:2f:3f:c6:c1:54:
        84:09:1a:2f:ef:61:09:ba:7c:31:fc:de:9c:15:2f:e4:f7:79:
        bc:82:4a:02:82:38:33:32:9d:48:8d:3a:53:85:fc:0c:5a:a4:
        7f:5a:e6:04:0e:1e:f5:b3:5e:86:1c:a9:84:47:a3:e8:65:1c:
        5a:87:3a:c8:55:3c:f6:0a:3d:1e:7f:78:a0:00:f8:bf:98:e0:
        6a:70:64:79:16:34:c1:4a:73:b7:0c:85:ec:82:00:27:0d:cd:
        af:55:2c:b4:5b:50:59:25:33:be:ad:68:d4:f1:53:8e:0f:7b:
        c0:61:9e:bc:0f:47:57:60:a2:e1:7f:b1:3e:e0:0e:ad:70:ea:
        89:f0:2b:48:1f:b1:36:dd:eb:43:6a:eb:eb:3b:0f:3c:24:70:
        93:43:b1:52
WARNING: no nonce in response
Response verify OK
certificate.pem: revoked
    This Update: Aug  1 01:09:01 2022 GMT
    Next Update: Aug  8 00:24:01 2022 GMT
    Revocation Time: Oct 27 21:38:48 2021 GMT
```

That we can make the OCSP request and received a response indicating that the certificate has been revoked, and then not proceed with the connection.

Q2 Suppression

I suppressed traffic using the `pf.config` file on MacOSX. AFter determining the IP for digicerts OCSP server, I suppresed that IP address using:

```
block drop from any to 142.251.32.35
```

If we suppress traffic to Digicerts ocsp verification url, then obviously we cannot verify the status of a certificate.

Our `scurl` applications stops working, as notable here:

```
mkdir: ./.scurl_tmp: File exists
OCSP Request Data:
    Version: 1 (0x0)
    Requestor List:
        Certificate ID:
          Hash Algorithm: sha1
          Issuer Name Hash: CF26F518FAC97E8F8CB342E01C2F6A109E8E5F0A
          Issuer Key Hash: 5168FF90AF0207753CCCD9656462A212B859723B
          Serial Number: 0E32CC48D16B1507813C2302D351BE89
    Request Extensions:
        OCSP Nonce:
            04101AB6FAA34EFF5D4D7FC67036A45546CC
```

It makes the request, but stalls on trying to get a response, and ultimately times out.

Chrome on the other hand continues as if the certificate were verified (or somehow knows that the certificate is NOT valid) Chrome could be doing this through caching.

This seems to also happen in firefox and in safari. Though I am not entirely sure what mechanism they each use.

After a bit of investigation, it appears that in addition to the OCSP url in the `Certificate Authority Information Access` extension for the x509 standard, there is another method for verifying a certificate information. That is a `.crt` file. Turns out this is the certificate file for the signer. So verifying the signer is another option.

It is also possible that the browser is using the CRL distribution points (which is also an extension in the x509 standard), or checking against the CRLs that are stored locally in the browser.

> We know from some research that Chrome uses CRLsets that are stored on the browser.