The homework instructions **for project 2.7** are included in the notes below. The notes are given for context from chapters 4 and 5 of out textbook. Find all of the highlighted LABS, and execute the openssl commands. There are 10 command lines to execute. Record the results or failure modes in a word.doc and submit to canvas. (NOTE: be careful with hyphens and underscores, or you will get syntax errors from openssl.)

**Chapter 4: Revocation and Invalidation**

**Revoking Certificates:**

* Several protocols have been developed to cope with Revoking and issuing new certificates.
  + Certificate Revocation Lists, (CRLs)
  + Online Certificate Status Protocol, (OCSP)
  + OCSP Stapling.
  + Current systems may use any or all of these.

**ONE, Certificate Revocation Lists:**

* The first or traditional way for revocation was/is CRLs.
* CRLs were created in the 1990s when certificates were expensive, CAs were few, and very few web sites needed encrypted connections.
* Like so many early internet protocols, CRLs did NOT scale.
* Consider also that many CAs update the CRL only every 3 hours … others do not … (motto).
* Your clients might be vulnerable for up to 27 hours after you revoke a certificate.
* **LAB** openssl crl command:
  + **Openssl crl -text -inform DER -noout -in myCA.crl (page 104)**
* **Some CAs have abandoned CRLs in favor of OCSP.**

**Text

Description automatically generated**

[**https://blog.didierstevens.com/2013/05/08/howto-make-your-own-cert-and-revocation-list-with-openssl/**](https://blog.didierstevens.com/2013/05/08/howto-make-your-own-cert-and-revocation-list-with-openssl/)

**TWO, Online Certificate Status Protocol (OCSP)**

* The OCSP seeks to improve performance by querying the status of a single certificate rather than downloading the entire list of revoked certs.
* The OCSP responder answers with: Good / Revoked / or Unknown, as well as how long the client can cache the answer … (length of time that the response is valid.)
* It is important to check the full certificate chain, not just the end certificate.
* **LAB: openssl x509 -noout -ocsp\_uri -in fullchain.pem** [**http://r3.o.lencr.org**](http://r3.o.lencr.org)

**Text

Description automatically generated**

* **LAB: open s\_client -showcerts -connect mwl.io:443 | openssl x509 -noout -ocsp\_uri (105)**

**Text

Description automatically generated**

* And then testing one of the author’s certs.
* **LAB: openssl ocp -issuer chain.pem -cert cert.pem -text -url** [**http://r3.o.lencr.org**](http://r3.o.lencr.org)

Text

Description automatically generated

* Review output from (**page 106**)

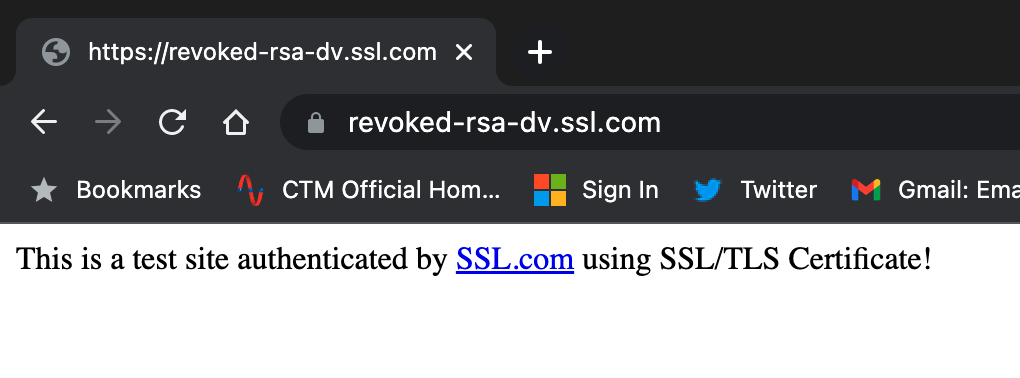
**Browsers Versus Revocation**

* TLS exists in many application, but the MOST VISIBLE are Web browsers.
* Firefox, Chrome, and Safari have fallen back to CRLs that are distributed as part of browser updates! (They each DO NOT use a common method:
  + Firefox uses OneCrl,
  + Chrome uses CRLSets,
  + Safari doesn’t give their method a name!
* Safari and Chrome do not check OCSP or Stapling by default // But you can enable them.
* Chrome’s CRLSets only contain sites that the Chrome team considers “important”
* It is a mess … so, many organizations provide deliberately revoked web pages where you can test your organizations browsers …
* **LAB**: visit ( <https://revoked-rsa-dv.ssl.com> ) with each of the 3 browsers, and compare results.
  + Firefox results …

Graphical user interface, text, application, chat or text message

Description automatically generated

* + Chrome results …



* + Safari results …

Graphical user interface, text, application, chat or text message

Description automatically generated

* Cut and paste results to a word.doc

**Chapter Five: TLS Negotiation.**

**Introduction: In the real world TLS Servers and clients must negotiate with each other at every connection, for example…**

* **Which algorithms can each side support?**
* **Which version of TLS?**
* **Each side can request or demand specific protocols.**
* **Which features are enabled or disabled?**
* **LAB: openssl s\_client -crlf -tls1\_2 -connect** [**www.mwl.mwl.io:443**](http://www.mwl.mwl.io:443) **(author site)**

**Text

Description automatically generated**

* **LAB: openssl s\_client -connect imap.gma.com:995 (gmail)**

****

**Certificate Validation:**

**Include the analogy of a traffic stop, the policeman asked for your driver’s license. There may be three different version of your license. The police officer checks to make sure that your license has not be revoked,**

* **Every TLS negotiation begins by validating all the certificates involved. TLS client only needs to find one valid path between the trust anchor and the host certificate.**
* **For TLS 1.2 and 1.3 the process looks identical.**
* **Follow the dialogue (pages 112 – 113) for the LAB above.**
* **TLS Failure Examples** 
  + **LAB: visit (** [**https://badssl.com**](https://badssl.com) **) // report on your findings from this site.**

**BadSSL.com is a website that was created to demonstrate the various security flaws and vulnerabilities that exist in the SSL/TLS protocols. Some of the basic findings of BadSSL.com include:**

1. **Certificate Errors: Some certificates are not issued by trusted certificate authorities and may not be verified, leading to a certificate error.**
2. **Mixed Content Warnings: Websites may contain both secure (HTTPS) and insecure (HTTP) content, which can lead to mixed content warnings in browsers.**
3. **Domain Name Mismatch: The domain name on the certificate may not match the actual domain name of the website, which can lead to a security warning in browsers.**
   * **And then:**
   * **LAB: openssl s\_client -verify\_return\_error -crlf incomplete-chain.badssl.com:443**

**Text

Description automatically generated**

* + **LAB: openssl s\_client -crlf incomplete-chain.badssl.com:433**

**Text

Description automatically generated**

* + **Cut and paste error messages to Google for more information…**
  + **Note be careful with ( \_ underscore and – hyphen ) when entering commands.**