SSL/TLS Web Server Security

Joint Techs, Fairbanks
Tuesday, July 14th, 2011, 4:10-4:30 PM

Joe St Sauver, Ph.D.
joe@uoregon.edu or joe@internet2.edu
Internet2 Nationwide Security Programs Manager
http://pages.uoregon.edu/joe/web-crypto

Format: This talk is provided in a detailed format to facilitate indexing by search engines, to insure accessibility for the hearing impaired, and to assist non-native English speakers who may view this presentation later.

Disclaimer: all opinions expressed are those of the author.

Note: This talk is based on a longer "TechBurst" webinar the author did for the REN-ISAC on June 30, 2011.

Why Worry About Web Security Now? Six Reasons

Reason #1: The Web Is A Common Bearer Service

- While dedicated clients using specialized network protocols were once common, these days virtually all enterprise network applications are accessed via a common bearer service -- **"everything is over the Web."**

- This is true for your users' email, calendaring and scheduling, campus administrative applications, GUI network device configuration, high performance computing (via web science gateways), and even campus ecommerce activities (whether that's buying a ten buck tee shirt as part of a departmental fund raiser or paying $10,000 in tuition for the term).

Reason #2: Web Apps Are A Prime Focus For Attacks

- Quoting from:
  http://www.sans.org/top-cyber-security-risks/summary.php

  Priority Two: Internet-facing web sites that are vulnerable.

  Attacks against web applications constitute more than 60% of the total attack attempts observed on the Internet. [...]

  Despite the enormous number of attacks and despite widespread publicity about these vulnerabilities, most web site owners fail to scan effectively for the common flaws and become unwitting tools used by criminals to infect the visitors that trusted those sites to provide a safe web experience.

  [Priority One? "Client-side software that remains unpatched."]

Reason #3: Many Education Web Sites Remain Vulnerable

- **"Most Websites Vulnerable to Attack, WhiteHat Study Says"***

  The average website has serious vulnerabilities more than nine months of the year, according to a new report [...]

  Heavily regulated industries like healthcare and banking have the lowest rates, yet 14 and 16 percent, respectively, of the sites in those industries had serious vulnerabilities throughout the year. [...]

  The education industry has the dubious honor of leading the category -- **78 percent of [education] sites** [...] **were vulnerable** [...]

---

* www.darkreading.com/vulnerability-management/167901026/security/application-security/229300525/most-websites-vulnerable-to-attack-whitehat-study-says.html (March 8th, 2011)

Reason #4: Some May Mistakenly Believe That The Sheer Presence of An "https" Prefix In A URL Equates to Overall Web Site "Security"

- Many users have been trained to check to see if web sites use "https" (SSL/TLS) before they trust personally identifiable information (such as credit card numbers) to a web site.
- SSL/TLS support *IS* an important part of securing a web site, but not all SSL/TLS implementations are the same, and just having some sort of SSL/TLS support, by and of itself, is not enough to make your website secure. (SSL/TLS support is "necessary but not sufficient," as mathematicians might say).
- **We need to "step up our game" when it comes to web site security in general (while also improving how we deploy SSL/TLS in particular).**
- Confusion on this point is similar to confusion about DNSSEC: while DNSSEC is needed to eliminate some DNS-related vulnerabilities, and it is an important thing for sites to do, DNSSEC does NOT fix all potential DNS vulnerabilities (nor does it pretend to do so). Similarly, SSL/TLS helps mitigate some web security vulnerabilities, but is not a magic pill

Reason #5: There Is (Appropriate!) Increasing Public Scrutiny Of Internet SSL/TLS Usage

http://www.eff.org/observatory

The EFF SSL Observatory

The EFF SSL Observatory is a project to investigate the certificates used to secure all of the sites encrypted with HTTPS on the Web. We have downloaded datasets of all of the publicly-visible SSL certificates on the IPv4 Internet, in order to search for vulnerabilities, document the practices of Certificate Authorities, and aid researchers interested the web's encryption infrastructure.

For the public, the slide decks from our DEFCON 18 and 27C3 talks are available, and you can also peruse our second map of the 650-odd organizations that function as Certificate Authorities trusted (directly or indirectly) by Mozilla or Microsoft.[1]

For the technical research community, our source code [2] as well as a MySQL database dump (August 2010 MySQL dump), the raw data (August 2010 raw data), and the August 2010 CSV database dump are available. You can also use the Observatory in an Amazon EC2 instance we created.

Please note that the data and code are not polished; patches and help are welcome. Questions can be asked on the project's mailing list or directed privately to <ssl-survey - at - eff.org>.

We are particularly concerned about the role and practices of Certificate Authorities (CAs), which are the organizations that can sign cryptographic certificates trusted by browsers. These certificates can contain statements like, "this public key belongs to EFF.org", "this public key belongs to yahoo.com, paypal.com and mozilla.com", or "this public key should be trusted to also act as a CA, signing certificates for other domains". Browsers trust a very large number of these CAs, and unfortunately, the security of HTTPS is only as strong as the practices of the least trustworthy/competent CA.

Before publishing this data, we attempted to notify administrators of all sites observed vulnerable to the Debian weak key

Reason #6: Internet2/InCommon Now Has Its Own Certificate Service

- You can read about it at http://www.incommon.org/cert/

- Because of that new cert service, those of us involved with Internet2 security have become motivated to look more closely at the "state of the practice" when it comes to certificate use, both for routine uses (such as for securing web servers), as well as for less common scenarios (such as deployment of "personal certificates")

- Let's now talk a little about SSL/TLS.

The SSL/TLS Protocols Go WAY Back...

- SSL is a relatively old technology (at least by World Wide Web historical standards), dating to 1994-1995 with the public release of SSL version 2.0* by Netscape...
- For context, Mosaic, the first popular graphical web browser, was created at NCSA and released in 1993.
- SSL/TLS has continued to evolve over time:
  -- 1996: SSL version 3.0
  -- 1999: TLS 1.0 (aka SSL 3.1) <-- *the latest broadly*
  -- 2006: TLS 1.1 (aka SSL 3.2) *adopted version,*
  -- 2008: TLS 1.2 (aka SSL 3.3) *believe it or not!*
- Unfortunately we've not keep up. **Many sites, including many universities and R&E sites, have weaknesses or exhibit known SSL/TLS-related vulnerabilities. Does yours?**

---

* SSL version 1 was reportedly never publicly released.

PLEASE CHECK *YOUR* "Secure" Web Servers

- **Check the domains/secure website(s) YOU care about at**

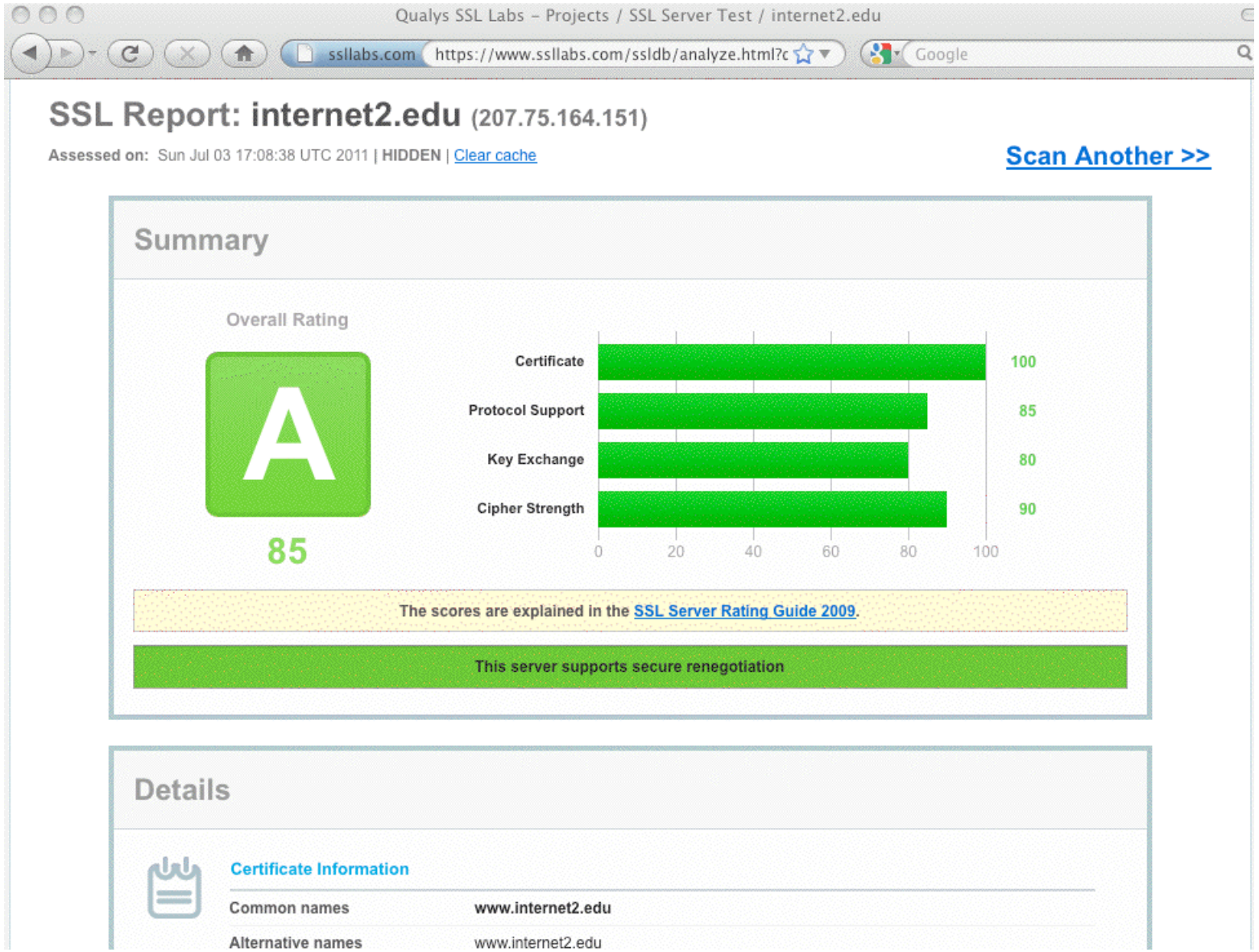
  https://www.ssllabs.com/ssldb/index.html

- Note that you can "hide" your scores if you're worried you might do badly!

- I've yet to see ANY SITE get a perfect "100" score, but most servers can pretty easily get "tidied up" to the point where they can get a nice solid score in the 85-88 range, high enough for an "A" grade on that assessment.

- I went ahead and checked nearly 120 edu domains just to get a sense of how the community was doing...

(Part of) A Sample SSL Labs Server Report



Higher Ed SSLlab Score Distribution for 119 Dot Edus, June 2011

score	Frequency	Percent	Cumulative Frequency	Cumulative Percent	
0	7	5.88	7	5.88	<-- F (score<20)
48	10	8.40	17	14.29	<-- D (score>=20)
52	29	24.37	46	38.66	<-- C (score>=50)
57	4	3.36	50	42.02	
60	1	0.84	51	42.86	
61	19	15.97	70	58.82	
62	1	0.84	71	59.66	
73	8	6.72	79	66.39	<-- B (score>=65)
76	1	0.84	80	67.23	
81	5	4.20	85	71.43	<-- A (score>=80)
84	1	0.84	86	72.27	
85	25	21.01	111	93.28	
86	1	0.84	112	94.12	
88	7	5.88	119	100.00	

Mean=62.8

Q3 (75th percentile)=85, Median (50th percentile)=61, Q1 (25th percentile)=52

Some Additional Higher Ed SSLlab Results...

Does the server permit SSL 2.0? (It shouldn't - SSL2.0 is known to be insecure):

NO	76 (63.87%)
YES	**43 (36.13%)**

Does the server do renegotiation securely? (insecure renegotiation is also bad)

YES	54 (45.38%)
BLOCKS ENTIRELY	18 (15.13%)
NO (VULNERABLE)	**47 (39.50%)**

What's the minimum cipher length acceptable to the server? (128 bit or better is good)

128 BIT	41 (34.45%)
168 BIT	1 (0.84%)
EVEN ANONYMOUS CIPHERS ARE OK	**2 (1.68%)**
40 bit	**63 (52.94%)**
56 bit	**12 (10.08%)**

Server cert signature length? (2048 bit or longer is now recommended)

2048 bit	53 (44.54%)
768 bit	**1 (0.84%)**
1024 bit	**65 (54.62%)**

And there's a lot more data out there if you look at the sites you're responsible for...

The First Very Basic Step: Make Sure Your Web Server Software Is Current

- There are two major Apache release trains, 1.x and 2.x
- While Apache 1.3.42 was released in February 2010 (and thus may feel relatively "current"), it was (and is) the final release in the Apache 1.x family.
- **If you're still using any 1.x version of Apache (and some people in higher ed *ARE*), or if you're using an early 2.x version of Apache, you should upgrade.** At the time I prepared these slides in late June 2011, the most recent production version of Apache 2.x was 2.2.19 (released May 22, 2011).
- If you're using Microsoft IIS/6 (and some people in higher education are), you should also be looking at upgrading to the current version of IIS, which is currently IIS/7.5

Obsolete Server Versions Observed In Edu SSLlab Results

apache (version not specified)	29 (24.37%)	
apache 1.3.26	1 (0.84%)	<-- circa June 2002!
apache 1.3.28	1 (0.84%)	
apache 1.3.37	3 (2.52%)	
apache 1.3.39	1 (0.84%)	
apache 1.3.41	1 (0.84%)	
apache 2.0.46	1 (0.84%)	<-- circa May 2003!
apache 2.0.50	1 (0.84%)	
apache 2.0.52	1 (0.84%)	
apache 2.0.52 (rh)	5 (4.20%)	
apache 2.0.54 (fedora)	1 (0.84%)	
apache 2.0.59	1 (0.84%)	
apache 2.0.63	1 (0.84%)	
[...]		
iis/6.0	13 (10.92%)	
iis/7.0	2 (1.68%)	

[plus some other really odd corner cases]

Tailor/Harden Your Now-Current Apache Install

- The as-shipped Apache config file will generally run fine as-is for a basic web server (although you should *at least* tailor it to have an accurate ServerAdmin email address).

- However are **many** additional things that you can and SHOULD also do via httpd.conf to help harden your server; some excellent starting suggestions are in:

  "20 Ways to Secure Your Apache Configuration,"
  http://www.petefreitag.com/item/505.cfm

- One of the most helpful things you can do is to install a web application firewall. A popular choice for Apache is mod_security (see http://www.modsecurity.org/) with the OWASP mod_security core rule set (from the same site)

You've Got More Work To Do

- Although you may now have a current web server installed and you may even be running mod_security, enabling SSL/TLS on that server requires you to obtain (or create) a cert, and to then configure the server to do SSL/TLS.
- Many operating systems will have a vendor web page or some other documentation walking you through the process of creating a certificate signing request (or even a fully "self-signed" certificate), while also helping you to enable mod_ssl (the Apache module that is normally used to enable SSL/TLS). For example, for Mac OS X, see: http://developer.apple.com/internet/serverside/modssl.html
- We're going to assume that you'll be using a package manager to install mod_tls, which means that it will automatically satisfy any unresolved dependencies (such as installation of Openssl)

The Process of Creating A Cert With OpenSSL

1) *Create a PEM-format 3DES-encrypted RSA server private key*

% openssl genrsa -des3 -out server.key 2048
% chmod 0400 server.key <-- protect your private key from being read

Note: pick a strong password and do NOT forget it!
Back up server.key (and your password!) somewhere safe!

2) *Create a PEM-format Certificate Signing Request*

% openssl req -new -key server.key -out server.csr

Note: when asked for your "Common Name," this MUST be the fully qualified domain name of your server!

That CSR could then be forwarded to a real certificate authority for signing, or we could sign it ourselves. For the purpose of this part of the discussion, we'll create our own "certificate authority" and issue and sign our own server certificate, purely for discussion purposes.

Creating Our Own "Certificate Authority"

1) *Let's create a 2048 bit key for your own "certificate authority"*

% openssl genrsa -des3 -out ca.key 2048
% chmod 0400 ca.key

Note: pick a strong password and don't forget it!

Back up ca.key (and your password for that key!) somewhere safe!

2) *Now create a self-signed "CA" cert*

% openssl req -new -x509 -days 365 -key ca.key -out ca.crt

3) *Now create and sign the server cert with the "CA" cert you made*

% openssl x509 -req -days 365 -in server.csr -out server.crt \
-CA ca.crt -CAkey ca.key -CAcreateserial

You can then copy server.crt and server.key and server-ca.crt into place in the locations specified in the httpd-ssl.conf file.

Additional Key Edits to Do In httpd-ssl.conf

In the default VirtualHost stanza, localize appropriately:

ServerName someserver.example.edu:443
ServerAdmin johnsmith@example.edu

Only do higher security ciphers, and only use trustworthy SSL Protocols:
SSLCipherSuite ALL:!aNULL:!ADH:!eNULL:!LOW:!MEDIUM:!EXP:+HIGH
SSLHonorCipherOrder on

 # SSL Protocol Support
SSLProtocol -ALL +SSLv3 +TLSv1

Point to the locations of the cert files:

SSLCertificateFile "/opt/local/apache2/ssl.keys/server.crt"
SSLCertificateKeyFile "/opt/local/apache2/ssl.keys/server.key"
SSLCertificateChainFile "/opt/local/apache2/ssl.keys/server-ca.crt"

What Are The Parameters in Those SSLCipherSuite and SSProtocol Lines?

-- See http://httpd.apache.org/docs/2.0/mod/mod_ssl.html#sslciphersuite

SSLCipherSuite ALL:!aNULL:!ADH:!eNULL:!LOW:!MEDIUM:!EXP:+HIGH

That forbids auth algorithms w/o authentication (!aNULL), forbids Diffie Hellman authentication (!ADH), forbids null cipher authentication (!eNULL), forbids Low and Medium strength ciphers (!LOW, !MEDIUM) and export ciphers (!EXP); and says the server should use High strength ciphers.

-- See http://httpd.apache.org/docs/2.0/mod/mod_ssl.html#sslprotocol

SSLProtocol -ALL +SSLv3 +TLSv1

That command disables SSLv2, an inherently insecure protocol that you should NEVER use (see RFC 6176, "Prohibiting Secure Sockets Layer (SSL) Version 2.0")

Looking for TLS 1.2 Support?

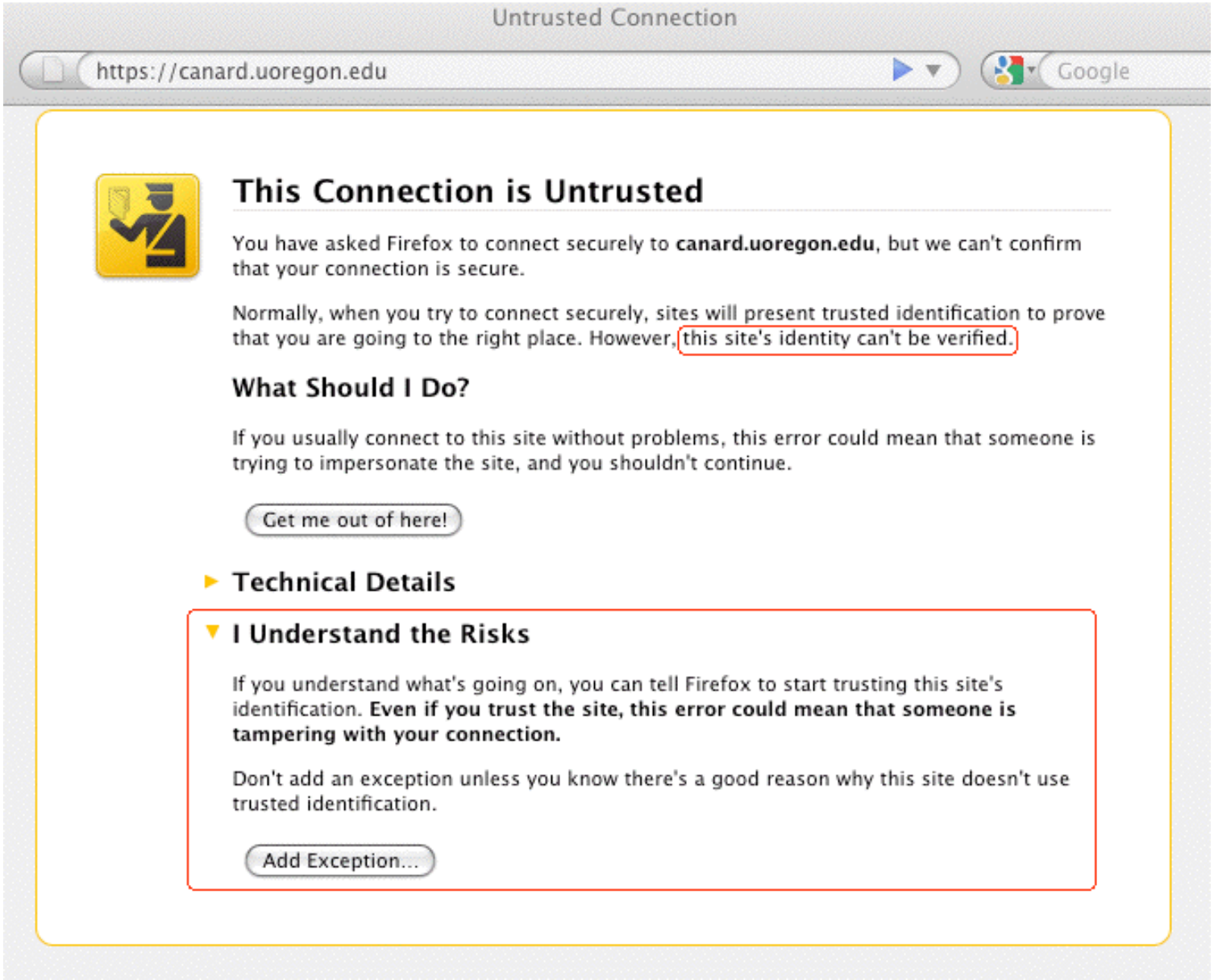
- OpenSSL supports TLS v1.0, but currently shipping production versions of OpenSSL **DO NOT** do TLS v1.1 (RFC4346, April 2006) nor TLS v1.2 (RFC 5246, Aug 2008), at least as of the time these slides were built.

- If you're an "enthusiast" and want support for TLS v1.1 or TLS v1.2, you may want to see the alternative TLS implementations mentioned at en.wikipedia.org/wiki/Comparison_of_TLS_Implementations (But is there a "mod_foocrypt" to easily integrate all of those alternatives? For gnutls yes, there is mod_gnutls, but in some other cases, no...)

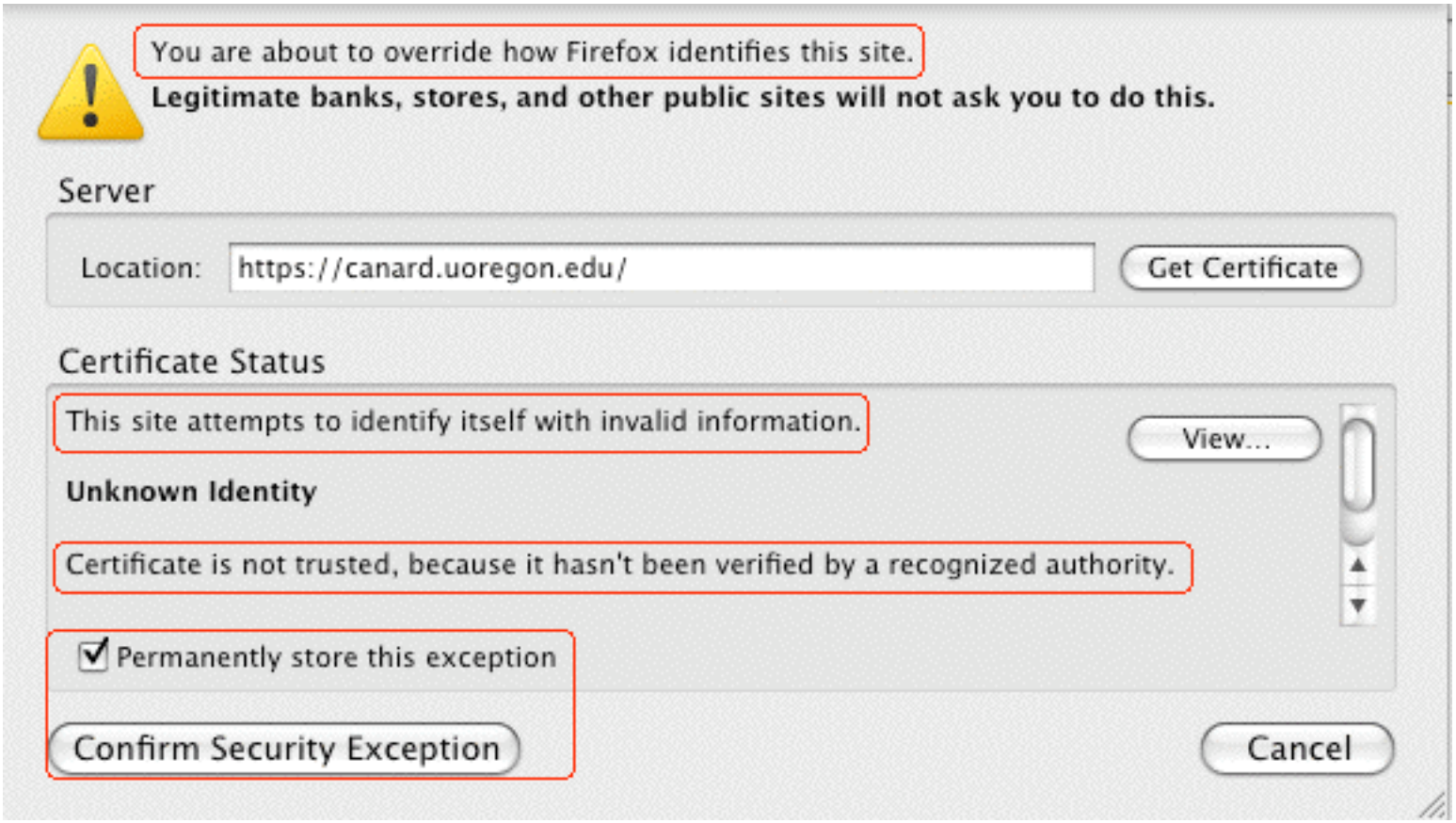
- Some TLS 1.2 implementations are also fairly exotic/experimental and may be thinly supported, tricky to successfully build on some operating systems, or lack other features (like compression support).

- Browser support for TLS v1.2 also remains regrettably uneven (en.wikipedia.org/wiki/Transport_Layer_Security #Browser_implementations). Kudos to MSIE and Opera in this case!

This Example Warning Is NOT An "Error"

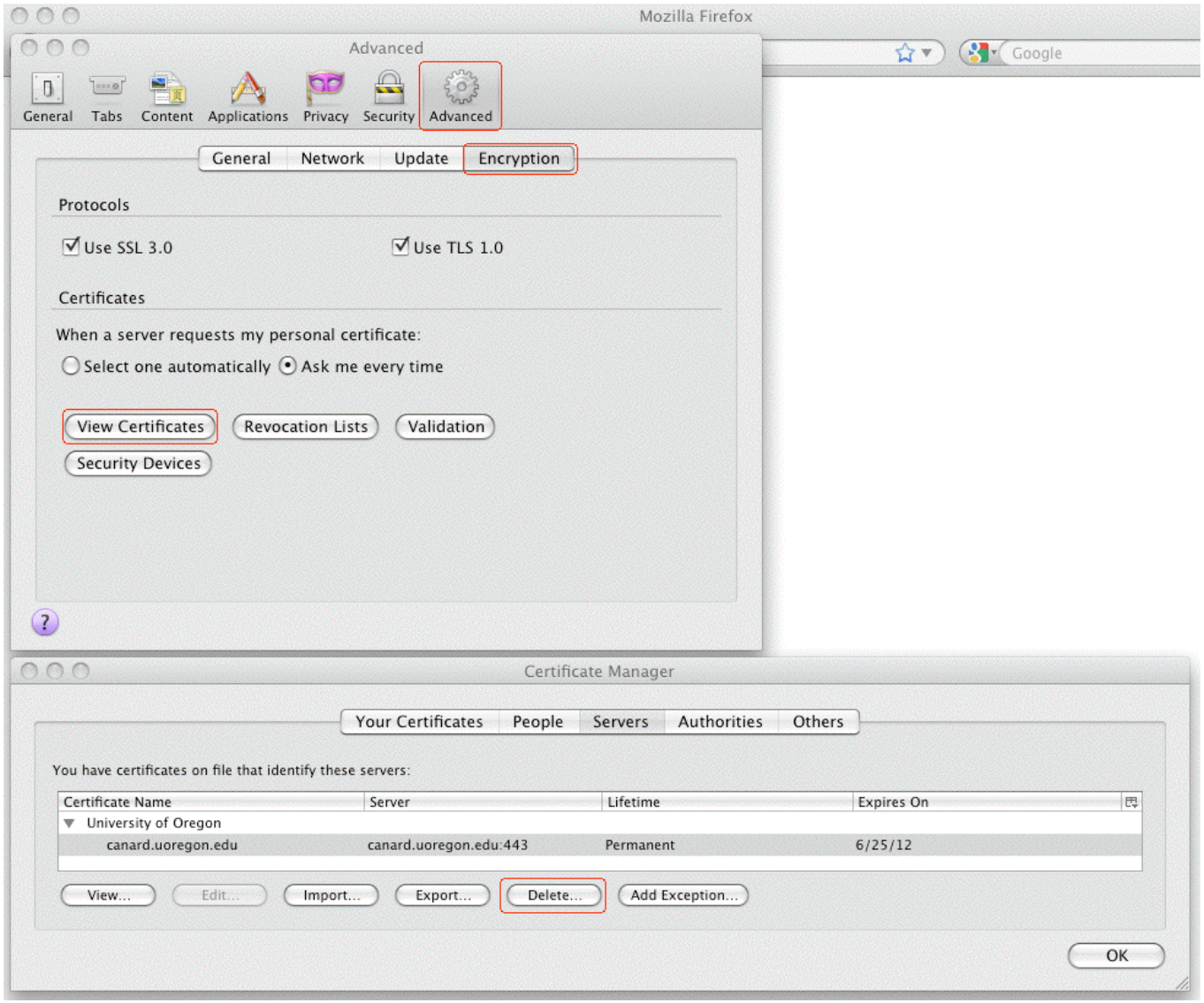


If You WERE to Click "Add Exception" (Doh!)



In spite of all those warnings, most users will, naturally, happily proceed to click on "Confirm Security Exception." **At that point, the SSL/TLS "trust" game is over for that server...**

What If You Want To Delete A (Mistakenly) Trusted SSL/TLS Server Certificate? In Firefox Preferences…



It Can Get Worse...

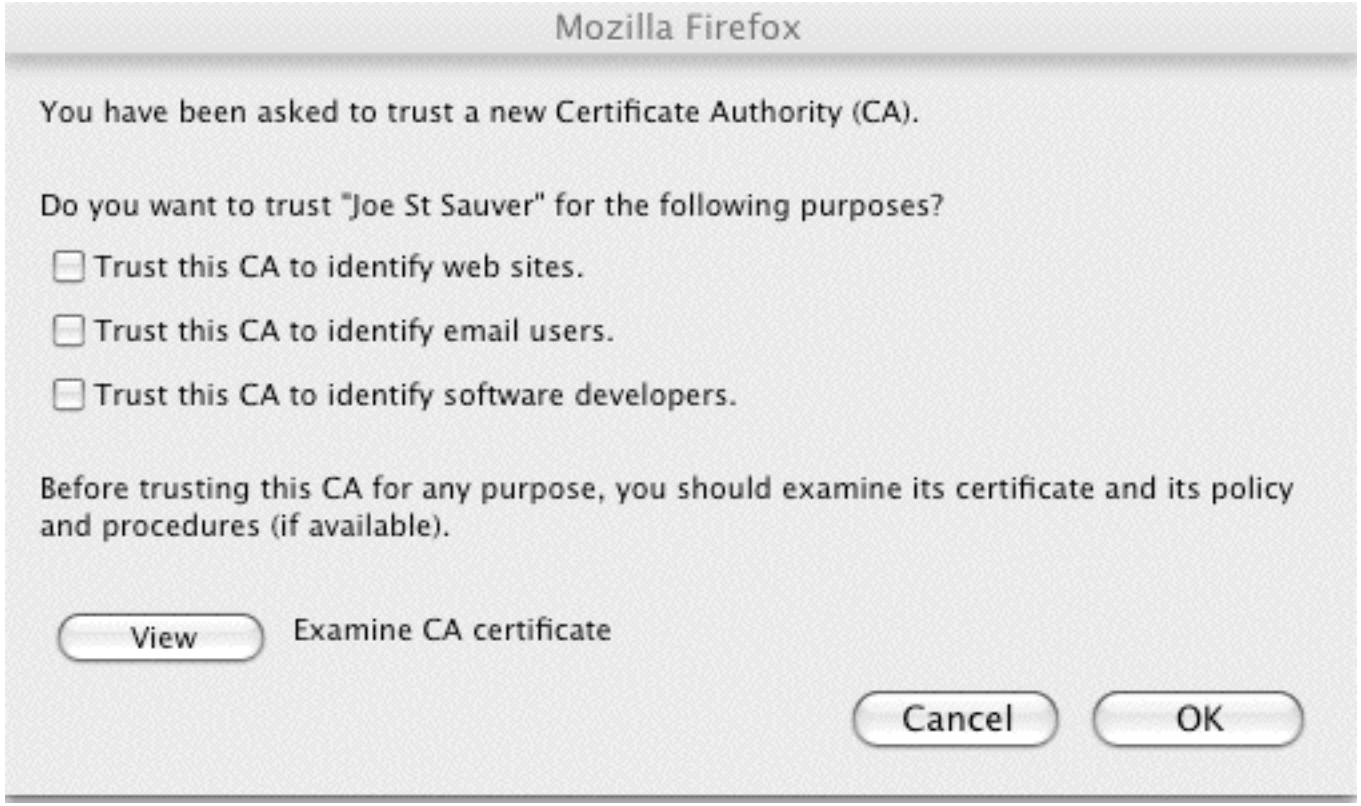
- I'm not going to give you an actual URL to click on, but let's assume that someone on the Internet did ask you to click on a link that pointed to a URL that looked like...

  http://www.example.com/my-ca.crt

  Would you do it? Would you click on that link? I think many people would - heck, people click on phishing URLs all the time, to say nothing of malware URLs, right? There's nothing that looks particularly evil about that link (I mean heck, it doesn't end in .exe or anything, right?)

- If someone did click on a link like that, they might see a popup dialog that looked like...

The Rather Matter-of-Fact Warning You See When You're Offered A New Certificate Authority



Note: Most users won't examine the CA certificate, or if they did, they typically won't understand/correctly interpret what they'd likely be shown. Most users have been trained to "always" just click "OK"

Compare That Quite Low Key New CA Warning Dialog To the Earlier Positively Shrill Self-Signed Cert Dialog

- On slide 22, we showed you the relatively in-your-face dialog box Firefox displays when you run into someone who's trying to get you to accept a self-signed cert. It was pretty shrill. Remember the little "passport inspector" logo and the "Get me out of here!" text?

- *Contrast that with what you just saw on the preceding slide. Given the unbounded destruction that trusting a random CA can impose, don't you think that the "Are you SURE you want to accept this new CA?" dialog should have a few more bells ringing and flashing lights going off???*

- In my opinion, that's a pretty matter-of-fact dialog box for such a potentially security-devastating decision!

What Could Happen? Man In The Middle (MITM) Attacks

- SSL/TLS is supposed to provide **end-to-end** encryption, all the way from your browser, all the way to the remote site's secure web server. When traffic is subject to a successful MITM attack, that ceases to be true. When someone manages to successfully conduct a MITM attack, they get between you and the server you're trying to securely communicate with, impersonating that real server.
- They (rather than the ultimate destination) can accept and decrypt your encrypted traffic. They can then view (and/or modify!) that traffic, before surreptitiously re-encrypting it via a second SSL/TLS session, and sending it on its way.
- If SSL/TLS works the way it is supposed to, it wouldn't be possible for you to be conned into trusting an imposter's system - the imposter wouldn't have the certificate it should have, signed by a trusted CA. If users decide to trust a new random CA, however, that model can fall apart

Shared Computers Can Be Very Vulnerable

- We're all familiar with shared computers – we have them in our homes, in our campus computer labs, in cyber cafes, in libraries, in hotel lobbies, at conferences, etc.
- If those systems aren't **COMPLETELY** locked down and **ROUTINELY** re-imaged to a known-good state after **EVERY USE**, a malicious (or clueless) user could accept a bogus certificate authority (it only takes a few seconds to do so), and then:
  -- via DNS changer malware, configure the system to use an untrustworthy recursive resolver ("DNS server"), thereby driving subsequent users to a web server of the malicious user's choice that will *seem* to be the secure and trustworthy destination they wanted
  -- alternatively, the attacker could just "securely" proxy all the user's https traffic (and snoop on it that way)

The Default Set of CAs in User Browsers

- Users have the discretion to **add** additional certificate authorities to their list of trustworthy CAs, as we just showed you. Obviously that's a huge potential risk.

- Users can also review the default list of as-shipped browser-trusted certificate authorities, and **delete** any CAs that they don't like (but few people do).

- **In most cases, user simply blindly trust those who create and distribute browsers to ultimately decide which CAs should be considered to be "trustworthy" by default.**

- There are some things about that that should make you unsettled.

Different Browser Vendors Trust Different Default CAs

- While you might expect all vendors to trust an agreed upon common set of commercial certificate authorities, that's not the case. (We'll leave comparing and diff'ing the various default CA lists, and speculating on the reasons for the differences between the various vendor lists, as an exercise for the reader). To get you started:

  -- Mozilla Included Certificate List
  http://www.mozilla.org/projects/security/certs/included/
  -- Opera Root Store
  http://my.opera.com/rootstore/blog/
  -- Windows Root Certificate Program Members
  http://social.technet.microsoft.com/wiki/contents/articles/3281.aspx

  Note: those lists can and do get "automatically" updated over time!

- You can also check the list of CAs your browser trusts by checking from within that browser. For example, in Firefox, go to Preferences --> Advanced --> Encryption --> View Certificates --> Authorities.

Should Each of Us Really Be Trusting All Default CAs?

- There have been many reports in the media about (potentially state-sponsored) cyber attackers aggressively targeting cutting edge intellectual property, such as new U.S. scientific discoveries or undisclosed inventions.
- We've all also heard repeated reports alleging that (some) foreign governments routinely conduct cyber surveillance of peaceful political and religious dissidents in the U.S.
- While I trust our government to abide by the rule of law (e.g., acquiring court orders for any interceptions they may conduct), I'm not sure I trust all foreign governments.
- Out of all the default certificate authorities in your web browser, could there be at least *one* CA that's under the influence or control of a foreign government? If so, we need to worry about so-called "Compelled Certificate Creation" attacks...

Compelled Certificate Creation Attacks

'www.schneier.com/blog/archives/2010/04/man-in-the-midd_2.html

April 12, 2010

Man-in-the-Middle Attacks Against SSL

Says Matt Blaze:

> A decade ago, I observed that commercial certificate authorities protect you from anyone from whom they are unwilling to take money. That turns out to be wrong; they don't even do that much.

Scary research by Christopher Soghoian and Sid Stamm:

> **Abstract:** This paper introduces a new attack, the *compelled certificate creation attack*, in which government agencies compel a certificate authority to issue false SSL certificates that are then used by intelligence agencies to covertly intercept and hijack individuals' secure Web-based communications. We reveal alarming evidence that suggests that this attack is in active use. Finally, we introduce a lightweight browser add-on that detects and thwarts such attacks.

Even more scary, Soghoian and Stamm found that hardware to perform this attack is being produced and sold:

> At a recent wiretapping convention, however, security researcher Chris Soghoian discovered that a small company was marketing internet spying boxes to the feds. The boxes were designed to intercept those communications -- without breaking the encryption -- by using forged security certificates, instead of the real ones that websites use to verify secure connections. To use the appliance, the government would need to acquire a forged certificate from any one of more than 100 trusted Certificate Authorities.

Secure Renegotiation: A More Mundane MITM Risk

- In 2009, it was discovered that SSL and TLS were vulnerable to insecure protocol renegotiaton, potentially enabling an entire class of MITM attacks against SSL/TLS (see http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2009-3555)
- RFC 5746 (February 2010) described a protocol-level fix for the insecure renegotiation, but many sites have neither blocked renegotiation entirely (something of a blunt weapon when it comes to addressing this issue), nor implemented secure renegotiation (typically by updating their web server AND SSL/TLS implementation).
- Remember: nearly 40% of all the higher ed web servers I checked with the SSLlabs tool remain vulnerable to this risk as of the time I made these slides.

There Can Be Certificate Management Issues, Too. A Simple Example: Your School (Probably) Uses More Than One Domain

- I'd be willing to *bet* that all of your schools use more than one domain.
- You may have a closely managed certificate management process for the one dot edu domain your school primarily uses, but I bet you also have a "hodge podge" of dot coms or dot orgs or legacy edu domains that are also in use.
- How do certs for THOSE domains get handled?
- Anyone who's on the domain's whois points of contact, or who can read common role accounts (root, postmaster, etc.) for that domain, may be able to request a domain validation certificate from some CAs, and in many cases I'll be you that they've already done so!
- Is that how you expected this to all work? :-;

Certificate Characteristics

- We also haven't talked about certificate "characteristics:"
  -- What **DURATION** certificates should you buy?
  -- Are **WILDCARD** certificates a good idea? They're sure convenient, but might they be TOO convenient?
  -- Do you need certs with **2048 BIT SIGNATURES?**
  (www.incommonfederation.org/cert/doc/2048-bit-Certificates.pdf)
  -- How about **EXTENDED VALIDATION** (so-called "green bar") certs? (They cost more from traditional CAs, but that may not be true if you purchase them from the InCommon certificate program)
  -- What about things like "Step Up" or "Server Gated Cryptography" (SGC) certs? Believe it or not, some schools **are** still using them (I wouldn't be). (Nice article on that topic at http://www.sslshopper.com/article-say-no-to-sgc-ssl-certificates.html)

Using Server Certs Beyond "Just" The Web

- Another interesting topic: certificates aren't a technology limited just to web servers. Have you considered using certificates to secure other campus services, too?

- For example, you can (and should!) use SSL/TLS wherever else you can, or wherever passwords are transmitted, such as for:

  -- SMTPS (secure opportunistic encryption of server-to server email, when both sides support it)
  -- IMAPS (secure user access to their email via dedicated clients such as Thunderbird or Outlook - they ARE logging in, after all, right?)
  -- Secure Submit (secure authenticated email submission by users)

Personal Certs

- We also don't really have much time left today to talk about personal certificates, but that's another potential "game changer" from the InCommon Certificate program.

- Personal certificates have had limited deployment to date in the U.S., except in the government (where the "Common Access Card" or "CAC" is ubiquitous), but that **may** be about to change.

- Sometimes trying a new technology like personal certs is just a matter of getting your "feet wet."

- For example, what about using personal certs for S/MIME secured email?

S/MIME: An Alternative to PGP/Gnu Privacy Guard

- While PGP/Gnu Privacy Guard has garnered some traction in the technical community, it's still pretty uncommon among non-technical friends and relatives.
- An interesting potential alternative to PGP/Gnu Privacy Guard is S/MIME. Many email clients have built-in S/MIME support, and in some ways key management for S/MIME is far easier than PGP or Gnu Privacy Guard . If you'd like to try using S/MIME on an ad hoc **TEST BASIS**, I've got a draft one pager with instructions for doing S/MIME with Thunderbird on the Mac out at:

  http://pages.uoregon.edu/joe/smime/

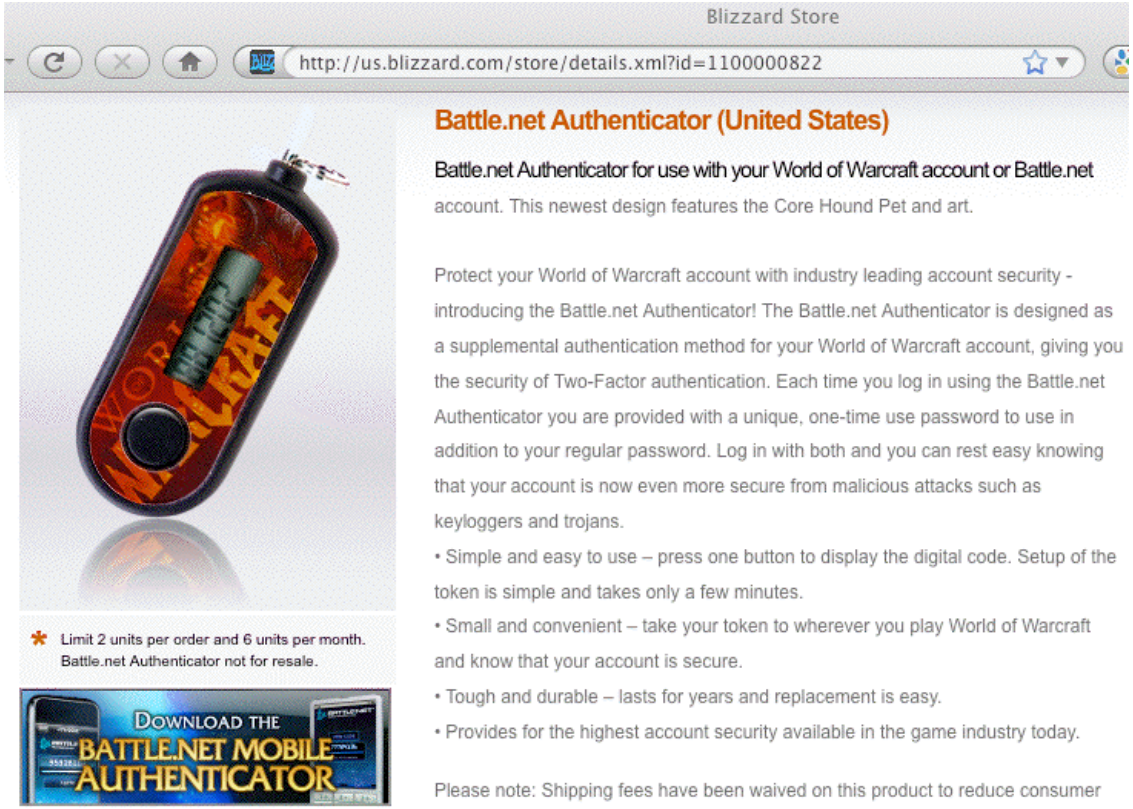
  That one pager's available in PDF and docx formats.
  Any feedback on that writeup would be appreciated.

Another Topic Related To Personal Certs

- While you *can* store personal certs in your browser on a dedicated laptop or workstation, it's far more secure to store personal certificates on a secure cryptographic token that you can carry with you at all times.
- I'd love to hear the community's experiences with secure ("FIPS-140 certified") hardware cryptographic devices that they may have used to hold personal certificates.
- Do the products you've tried work well with all platforms? (Mac, Linux, Windows, etc.?)
- How do you deal with portable devices that may not have an integrated USB port or smart card reader?
- Can users load (or reload) their own personal certs, or do they need to be loaded and administered centrally?
- Are hardware tokens for personal certs "priced right?"

A Benchmark Two Factor Authenticator, For Comparison

- We need to try to get to the point where we have secure personal-certificate-based hardware authentication devices that cost no more than my kid's $6.50 WoW authenticator…



Thanks For The Chance To Talk Today!

- Are there any questions?