SSL/TLS Certificates: Giving Your Use of Server Certificates A Hard Look

Joe St Sauver, Ph.D.

joe@internet2.edu or joe@uoregon.edu

InCommon Certificate Program Manager and

Internet2 Nationwide Security Programs Manager

Internet2 Member Meeting

4:30-5:30 Eastern, October 3rd, Room 305A

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

Note: Portions of this talk were previously presented by the author as a REN-ISAC Techburst session on June 30th, 2011.

I. Introduction

Where I’m "Coming From"

- ***Full Disclosure:*** I want folks to know that my responsibilities now include serving as InCommon’s Certificate Program Manager, in addition to my continuing responsibilities as Internet2’s Nationwide Security Programs Manager.

- If that potentially "biases" my interest in certificate-related security topics, well, you’ve been officially advised. :-)

This Is <u>Not</u> A Sales Pitch

- ***This is NOT a "Sales Pitch:"*** Yes, InCommon does run a Certificate Program for the community, and many of you participate in that program. Welcome today, and thank you!

  If you're not currently participating, but you buy a lot of certificates, try figuring out how much you spend on them. Compare that to http://www.incommon.org/cert/cert_fee.html

  If it makes sense for you to participate, we'd love to have you become part of the program too. If it doesn't, that's also cool. Obviously you should do what makes sense for your school.

Today's Session

- ***Session Emphasis:*** For the most part, I'd like this session to stay focused on technical/general certificate-related issues. If there are business/admin questions that relate specifically to the InCommon Certificate Program, I'd be happy to visit with you about those, but let's do that <u>after</u> today's session.

- ***Session Format:*** I'd like this to be an interactive session, so although I've prepared material to go over, feel free to chime in if you've got comments on any of the topics I raise – the value of this session will increase if everyone is actively engaged.

How This Talk Relates to Other Member Meeting Certificate-Related Sessions

- *Today's Talk* **--** "What are the security issues we face when it comes to hardening and improving the way we run existing SSL-enabled web servers?" That is, "How can we run our existing web servers more securely?"
- ***Leveraging Certificates for Improved Security (Tuesday October 4th, 1:15-2:30) --*** "Are we using certificates as broadly as we can? Or are there additional/new ways we could or should use SSL server certificates to improve our security? And what new frontiers are opened by use of client (personal) certs?"
- ***PKI BOF (Wednesday October 5th, Noon-1PM) --*** An interactive community-driven session, lead by Jim Jokl of U Virginia. Topics include the recent two factor auth survey and client certificate installers among others. Please attend!

Bringing Everyone To A Common Foundation

- Let's begin by talking a little about web security.
- I know that the material I'm going to begin with will be review for many of you; unfortunately the material that may be review for <u>you</u> probably won't be review for others. Since folks may be sharing this talk broadly, with a potentially diverse audience of folks, I want to get everyone to a common level before we move forward.
- I appreciate your patience for a few slides. This talk will pick up technical "velocity" as we move along, although I'm going to try to keep this talk approachable for everyone.
- Anyhow, our first questions are, and probably must be, **"Why are we particularly interested in web site security?"** and **"Why focus on this issue <u>now</u>?"**

Factor 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: (almost) "**everything is over the Web.**"
- This is true for your users' email, their calendaring and scheduling, campus administrative applications, 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).
- When web applications involve sensitive data (such as account usernames and passwords, FERPA- or HIPAA-covered data, or PII such as credit card numbers), that web activity will normally occur over a SSL/TLS-secured connection.

Factor 2: Web Apps Are A Prime Focus For Attacks

- 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.** These vulnerabilities are being exploited widely to convert trusted web sites into malicious websites serving content that contains client-side exploits. Web application vulnerabilities such as SQL injection and Cross-Site Scripting flaws in open-source as well as custom-built applications account for more than 80% of the vulnerabilities being discovered. 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."]

Factor 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** [...]

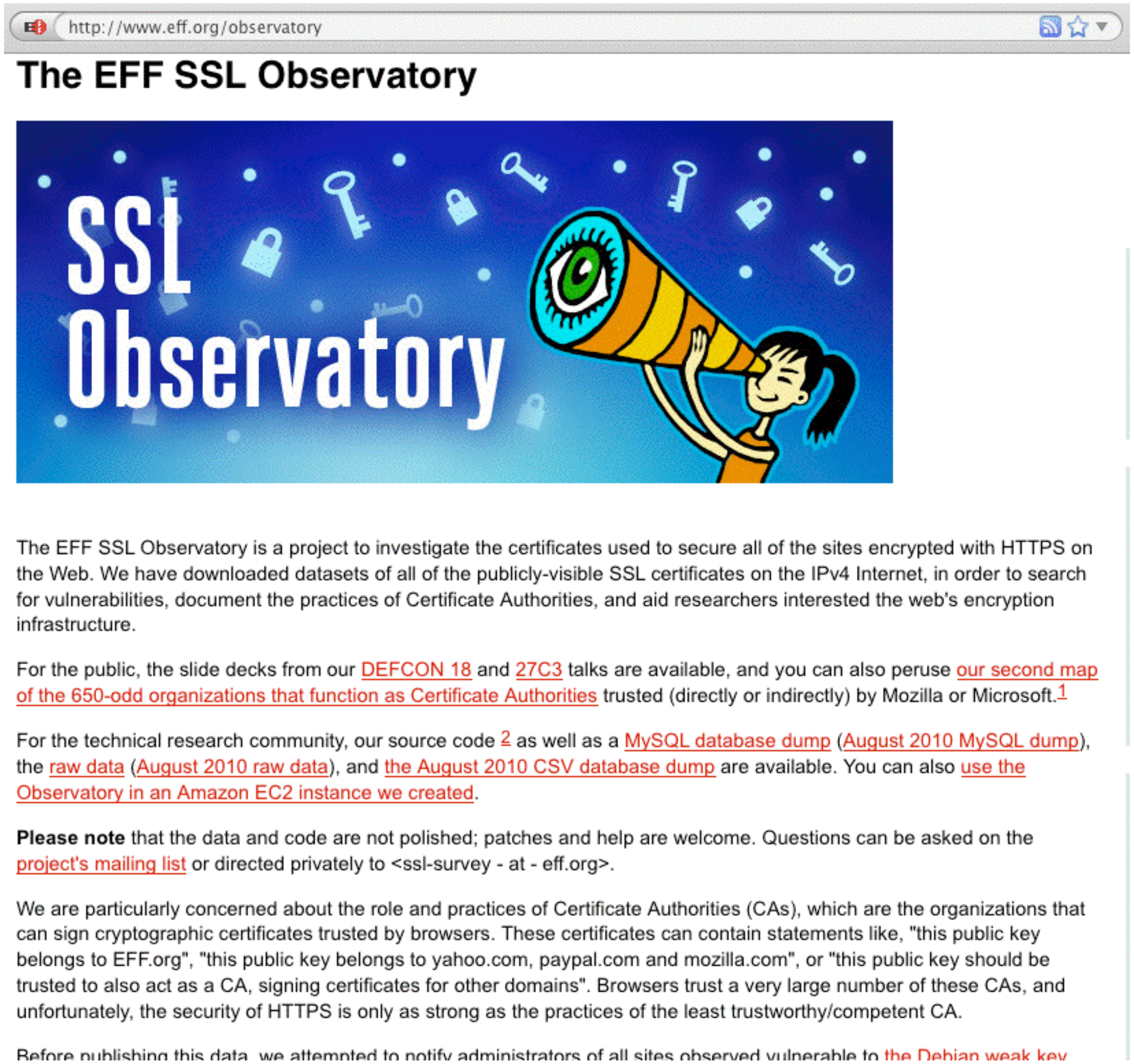
----

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

Factor 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

Factor 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

Factor 6: Internet2/InCommon Now Has Its Own Certificate Service

- The InCommon Certificate Service offers Certificate Service subscribers unlimited certificates, including unlimited SSL certificates and even unlimited extended validation certificates for one flat fee.
- 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 certs")
- That said, let me emphasize that the opinions expressed in this talk represent my own point of view, and not necessarily those of InCommon or Internet2, the University of Oregon, or any other entity.

Bottom Line

- We think the time may be ripe for many sites to improve their certificate-related practices, particularly as they pertain to their web site.
- We also think that it may be time for the community to begin to think about other possible uses for certificates, such as two factor authentication & S/MIME signed email.
- We hope you share our interest in these topics, and we view your participation in this session as a very positive tangible sign in that regard.
- We also hope you'll freely share these slides with your web server system administrators and other campus personnel who may be interested.
- Now that we've briefly established some web-related introductory motivational material, let's also quickly talk a little about SSL/TLS.

- But first, are there any burning questions so far? (We'll try to remember to check for questions at the end of each major section of this talk)

II. A Quick Hand-Waving Introduction To SSL/TLS

What Is SSL/TLS?

- The "Secure Socket Layer" ("SSL") and "Transport Layer Security" ("TLS") protocols are cryptographic technologies that are used, along with certificates, to help secure e-commerce websites and other Internet resources.
- SSL is a relatively old technology (at least by Internet 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 has continued to evolve over time:
  -- 1996: SSL version 3.0
  -- 1999: TLS 1.0 (aka SSL 3.1) <-- *the latest "universally supported" version, believe it or not!*
  -- 2006: TLS 1.1 (aka SSL 3.2)
  -- 2008: TLS 1.2 (aka SSL 3.3)

* SSL version 1 was reportedly never publicly released.

"So Tell Us About The Technical Differences Between the Versions of TLS, and How the TLS Handshake Process Works, and the TLS Record Format and…"

- No.

- While it is sometime considered *de rigueur* to do a "deep dive" with state diagrams and record layouts as part of a technical briefing, we don't have time to cover that today, and frankly you really don't need to know the protocol level details for our purposes.

- If we were doing a whole term-long class devoted to cryptography, that would be a different matter, but our time together today is short and I want to focus on stuff that's important from an operational security point of view. For example, what does SSL/TLS really do for us?

What SSL/TLS Does For Sites and Users

- By using SSL/TLS secured web sites, site administrators and their users get three potentially quite useful things:

  -- network traffic gets **protected from eavesdropping**
  -- network traffic gets **protected from tampering,** and
  -- users get **protected from accidentally going to a look-alike counterfeit site** (assuming the SSL/TLS certificate being used has been issued by a source that adequately validates the identity of the party to whom that certificate has been issued, and some other conditions are also satisfied)

- A tremendous amount of detail underlies those three fundamental objectives. You'll be able to see this if you do bother reading the SSL/TLS protocol-level RFCs.

By the (RFC) Numbers

- **RFC 2560**, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP," http://tools.ietf.org/html/rfc2560
- **RFC 5246**, "The Transportation Layer Security (TLS) Protocol, Version 1.2," http://tools.ietf.org/html/rfc5246
- **RFC 5280**, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," http://tools.ietf.org/html/rfc5280
- **RFC 5746**, "Transport Layer Security (TLS) Renegotiation Indication Extension," http://tools.ietf.org/html/rfc5746
- **RFC 5878**, "Transport Layer Security (TLS) Authorization Extensions," http://tools.ietf.org/html/rfc5878
- **RFC 6066**, "Transport Layer Security (TLS) Extensions: Extension Definitions," http://tools.ietf.org/html/rfc6066
- **RFC 6176**, "Prohibiting Secure Sockets Layer (SSL) Version 2.0," http://tools.ietf.org/html/rfc6176
- *Plus* bits and pieces in other RFCs and errata to most of the above... Note that these documents are NOT light/easy reading.

Stating the Obvious

- Many users/system administrators/security people **never have (and never will!)** read and internalize those RFCs, in part because understanding cryptographic protocols often require a degree of comfort with advanced mathematics.
- If you do want to read at least <u>a little</u> about SSL/TLS, Wikipedia actually has some nice introductory articles:

  en.wikipedia.org/wiki/Transport_Layer_Security
  en.wikipedia.org/wiki/Comparison_of_TLS_Implementations

- Fortunately, you really don't need an in-depth understanding of SSL/TLS protocols if you're not doing protocol-level development work. There's an active community of very well-regarded cryptographers and coders that are "carrying the water" for us in this area.

Practitioner-Level Crypto

- Practically speaking, for most practitioners, SSL/TLS "is" what Apache 2.x (and OpenSSL) say it "is."
- Why? As of June 2011, Apache currently has a ~65% market share compared to its next-closest competitor, Microsoft, at roughly 17%. See http://news.netcraft.com/archives/category/web-server-survey/
- Given that level of market dominance, we will largely focus on Apache (running on Unix) when we discuss web servers during the remainder of this talk.

- **Because many SSL/TLS issues come back to how Apache was installed and configured, let's now review how one actually does that installation and configuration.**

"But Joe! We <u>already</u> know how to install and configure a web server! You're wasting our time!"

[or alternatively]

"Why are you telling *us* how to install Apache??? We're CIOs or network engineers, *not* sysadmins!"

A Quick Reality Check

- **I don't want to point fingers at any particular site.** Everyone's doing the best that they can with the resources they have available. Unfortunately, though, sometimes things just aren't where they need to be.
- When I checked a sample of higher ed institutions with a popular SSL site checking tool from Qualys SSLLabs, I empirically observed higher ed sites that were "all over the map" when it came to their web server security.
- **I encourage YOU to check the 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'll do badly!)
- If you get a 100% score on that evaluation, I apologize in advance for wasting your time, however, if your site or sites gets a lower mark, maybe we should take a couple of minutes to review how to install Apache?

The Higher Ed SSLlab Score Distribution for 119 Dot Edus

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 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)

40 bits	**63 (52.94%)**
56 bits	**12 (10.08%)**
128 bits	41 (34.45%)
168 bits	1 (0.84%)
even anonymous ciphers OK	2 (1.68%)

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

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

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

(Please) Don't Shoot The Messenger

- I'm NOT trying to prove that I'm "smarter" than anyone else or that anyone's done a "bad job" with their web site.
- I'll freely concede that EVERYONE probably knows more about rolling out both regular and secure web sites than I do.
- On the other hand, I do want people to make an informed objective assessment of where their web sites are at, and to have some concrete ideas for how they might be able to improve them.
- I *would* like us to work on this cooperatively, as a community.
- If I had to describe ONE THING that I'd like you to do after today's session, it would be to check and fix any security issues with your school's secure web server(s).

III. Installing Apache

Apache 1.x vs. Apache 2.x

- 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 you're using anything other than the latest production 2.x release, you should upgrade (unless some "application-related constraint" makes this "impossible" [cough]). At the time I updated these slides in late September 2011, the most recent production version of Apache 2.x was 2.2.21.
- To see what version you are actually running, on most Unix systems look for the full path of the httpd that's running in the output from

```
% ps auxw | grep http              (some sites need ef instead of auxw)
```

For example, your httpd might be at /opt/local/apache2/bin/httpd

You can then see what version you're running by saying:

```
% /opt/local/apache2/bin/httpd -version
```

Versions Seen In Higher Ed SSLlab Results... Are All Secure?

apache (version not specified)	29 (24.37%)
apache 1.3.26	1 (0.84%)
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%)
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%)
apache 2.x	(omitted here)
[...]	
iis/6.0	13 (10.92%)
iis/7.0	2 (1.68%)
iis/7.5	2 (1.68%)

[plus some other really odd corner cases]

Beware Multiple Parallel httpd Installations

- Some of you may wonder why I bothered to have you check to see the full path for the httpd you're running.
- The answer is that it can (unfortunately) be quite common for a system to have MULTIPLE parallel httpd installations, and the version that you see by default from an interactive terminal session may (or may not) be the same version that's <u>currently</u> running or the same version that's <u>normally</u> launched at boot time (due to path issues, etc.)
- While it may be tempting to dismiss any installations in "wrong" places as "stupid," different distros may put the emphasis on different things (e.g., limiting "contamination" to the minimum number of file systems, obtaining the best system performance, protecting critical file systems from accidentally filling up, preserving a still-required vendor-pre-installed version, isolating sensitive config files, etc.)

Some Default File System Layouts For Apache

- A nice summary of some (but not all) Apache file system layouts can be found at

  http://wiki.apache.org/httpd/DistrosDefaultLayout

- Just to make EVERYONE equally unhappy, we'll use the default file location /opt/local/apache2 , which isn't used by any of the major vendors mentioned in the preceding file.

- Adjust the filespecs I show in the slides ahead according to the layout that your distro/installation uses.

Your Pre-Installed Version of Apache

- Because of its inherent modularity, potentially large number of dependencies, and differing file system layouts on different operating systems, Apache and related bits and pieces can sometimes prove to be a complex product to build from sources, install, and maintain.
- Fortunately, many popular operating systems come with a version of Apache pre-installed by default.
- On the other hand, that pre-installed version of Apache may lag the latest release (even after you apply all vendor updates), or lack a feature you need, or come statically built with features you don't need.
- You may thus want to (re) install the latest version of Apache even if there's a vendor version already installed.

Using a Package Manager or Port Tool

- One nice alternative to installing from scratch is to use a "package manager" or a "port tool" to install a professionally prepared port of Apache.
- Going this route saves you the pain of figuring out any tricks you may need to know in order to build Apache from scratch for your platform.
- Using a package manager or port tool will also make it to easy to stay patched up-to-date in the future.
- Unfortunately, each package manager/port tool is a little different when it comes to installing Apache.
- **We'll illustrate installation of Apache on a Mac with Mac ports (hey, we had to pick something, right?)**
- Begin by installing macports on your Mac OS X system if you don't already have it installed (see http://www.macports.org).

Example Apache Installation Using Mac Ports

- Once you have Mac Ports installed, you can install Apache by saying:

```
% port search apache        <-- find the package we want
% su                        <-- su doesn't work on your Mac? See
                            http://support.apple.com/kb/HT1528
# port install apache2
```

  (This will install apache2 and also recursively install any dependencies (such as apr, apr-util, expat, **openssl**, pcre, perl5, etc.) if needed).

```
# port load apache2
# launchctl load -w /Library/LaunchDaemons/org.macports.apache2.plist
```

  (This will set up this version of Apache to be the one that's run/used)

- You might also need to punch a hole in your firewall rules to expose your web server to the world (you will likely be automatically prompted to do so on most Macs). Note: do NOT go to System Preferences --> Sharing --> Web Sharing in an effort to allow httpd, you will end up launching Apple's default apache2, not the Apache you just installed!

Tailor httpd.conf

- The Mac Ports version of Apache ships with a basic httpd.conf config file at /opt/local/apache2/conf/httpd.conf
-
- FWIW, the as-shipped Apache config file will generally work fine as-is for a basic web server (although you should tailor that file with your favorite editor (vi, emacs, etc.), to **at least** have an accurate ServerAdmin email address).

- You should know, however, that there are **many** additional things that you can 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

(Optional) Installing mod_security2

- mod_security is a Web Application Firewal (WAF) that you can run to harden your Apache installation. While very helpful, unfortunately, many sites do not use it. To install it using Mac Ports, say:

  ```
  # port search mod_security2
  # port install mod_security2
  ```

- Edit /opt/local/apache2/conf/httpd.conf to include:

  ```
  LoadFile /opt/local/lib/libxml2.dylib
  LoadFile /opt/local/lib/liblua.dylib
  LoadModule security2_module modules/mod_security2.so
  ```

- You'll need to create and tailor a mod_security.conf file alongside your httpd.conf file (I got my starting mod_security.conf from the mod_security source files available at www.modsecurity.org). You will also need to retrieve and install appropriate mod_security rules, such as the Core Rule Set

(Optional) Installing mod_security2 (Continued)

- *Retrieve and install the mod_security core rule set:*

```
# mkdir /opt/local/apache2/conf/crs
# cd /opt/local/apache2/conf/crs
# wget "http://sourceforge.net/projects/mod-security/files/\
modsecurity-crs/0-CURRENT/modsecurity-crs_2.2.0.tar.gz/download"
# gunzip modsecurity-crs_2.2.0.tar.gz
# tar xfv modsecurity-crs_2.2.0.tar
# cd modsecurity-crs_2.2.0
# mv * ..
# cd ..
# rmdir modsecurity-crs_2.2.0
# more INSTALL
```

<-- *DO* what's described in here! :-)

- And be sure you have required config files included in httpd.conf:

```
<IfModule security2_module>
     Include conf/modsecurity.conf
     Include conf/crs/modsecurity_crs_10_config.conf
     Include conf/crs/activated_rules/*.conf
</IfModule>
```

Make Some Sort of Home Page For Your Web Server

- The httpd.conf file will tell you the location for your web server's document root; in our case it is /opt/local/apache2/htdocs

- cd to that directory, then create an index.html file (using vi, emacs, or your favorite editor), so the web server has something to display:

```
<HTML>
someserver.example.edu
```

- Make sure that file's readable by all:

```
# chmod a+r index.html
```

Start Apache

- You can then launch Apache:

```
# /opt/local/apache2/bin/apachectl start
```

Check To Make Sure Everything's Okay

Check to see if there are httpd's running (you will typically see several pre-spawned and ready-to-go, that's normal):

```
# ps auxw | grep httpd
```

If there aren't any httpds, check the log files for possible errors:

```
# tail -f /var/log/system.log              <-- ctl-C to interrupt
# tail -f /opt/local/apache2/logs/error_log
```

Everything looking okay? Now try connecting from a browser by plugging in the address of your server in the browser's address bar:

http://someserver.example.edu/

If you see the home page you created on the previous slide, you've got Apache running!

Some Random Thoughts On Log Files

-- Do you normally review your syslog and web logs? Do you think you SHOULD be paying (more) attention to your syslog and web log files?

-- Who's responsible for doing that review? Your web person? Your sysadmin? A security person?

-- How do you do it? Is there a log analysis tool you use?

-- What do you look for?

-- What do you do if you see anomalies (if anything?)

-- Have you considered secure centralized logging? (syslog-ng, etc.?)

IV. Enabling SSL/TLS On Apache2 With mod_ssl

You've Got (Still) More Work To Do

- You have a web server installed and running, h**owever, it's NOT a SSL/TLS secured web server.** Enabling SSL/TLS on that server requires you to obtain (or create) a cert, and 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 "self-signed" certificate and enabling mod_ssl (the Apache module that is normally used to enable SSL/TLS).

- For example, for OS X, see: http://developer.apple.com/internet/serverside/modssl.html

OpenSSL

- The material we're going to show you on the following slides assumes you have the latest version of OpenSSL installed (normally OpenSSL will automatically get installed as part of installing Apache, as an Apache dependency).

- Because OpenSSL does all the "heavy lifting" for our crypto, **we want to make sure that it's completely patched up-to-date.** As of the date this presentation was updated, that implies running OpenSSL 1.0.0e

```
% openssl version
OpenSSL 1.0.0e 6 Sep 2011
```

  [FWIW, Some package manager/port operations may not yet have a packaged version of this most recent release]

The Process of Creating A Cert With OpenSSL

1) Make a working directory and cd down into it:

```
% mkdir KeyGen
% cd KeyGen
```

2) *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
```

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

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

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

<u>Note</u>: when asked for your "Common Name," this must be the fully qualified domain name of your server!

For now, omit entering a challenge password / optional company name

"What's PEM and 3DES and RSA and..."

Besides the math that may be involved, one of the things that tends to discourage some people when they begin working with cryptographic apps is the amount of jargon involved (sorry about that!).

For example, on the preceding page, "PEM" stands for "Privacy Enhanced Mail" (even though what we're working on has nothing to do with mail). PEM format files are "base 64 encoded" text files (unlike some other non-printable binary format files). As text files, PEM-format files can easily be copied or transfered just like any other text file. (See en.wikipedia.org/wiki/X.509#Certificate_filename_extensions)

"3DES" stands for Triple DES, a common algorithm for encrypting content. See http://en.wikipedia.org/wiki/3DES "RSA" is yet another cryptographic algorithm. See http://en.wikipedia.org/wiki/RSA

Note that you do NOT need to understand the mathematical subtleties of these algorithms to successfully use SSL/TLS.

"Self-Signed" Vs. "Signed by a Real CA"

At this point, however, there IS one critical distinction that you do need to understand, and that's the difference between a self-signed cert, and a cert that's been signed by a real certificate authority.

You can create your own "certificate authority," and use that "CA" to sign your own certificate, OR you can request that a real (e.g., widely accepted) certificate authority issue and sign your certificate.

For the purpose of this part of the discussion, we'll create our own "certificate authority" and issue and sign our own server certificate.

Note: our creation of a CA certificate is being done as part of this talk as an exercise/example. I do NOT meant to imply that anyone can or should attempt to create a "trustable" CA this way!

For that reason, I'm going to put "CA" in quotes while we're talking about anything associated with our "self-made" "CA"

Creating Your 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
```

<u>Note</u>: 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
```

Now let's copy those files into place...

Moving The Certs and Key Files Into Place

```
% su
# mkdir /opt/local/apache2/ssl.keys
# cp server-ca.crt  /opt/local/apache2/ssl.keys/server-ca.crt
# cp server.crt  /opt/local/apache2/ssl.keys/server.crt
# cp server.key  /opt/local/apache2/ssl.keys/server.key
```

The server's private key is password protected. This means that you'd need to supply the password for that cert as part of the startup sequence. If you can't supply that password at startup, you're S-O-L. Many server admins therefore routinely strip the password from their server's private key, even though that reduces its security:

```
# cd /opt/local/apache2/ssl.keys
# cp server.key server.key.original
# openssl rsa -in server.key.original -out server.key
# chmod 0400 server.key      <-- IMPORTANT, Don't Forget To Do This!
```

Badness Inherent in That Process

There's a lot of inherent badness in the process you just saw, besides just stripping the password from the server's private key. Let me just mention a few examples:

-- when you created your server's certificate request you supplied a bunch of information; it never got validated by anyone (except yourself); ditto for the "CA" cert. **The "identities" associated with those public keys should NOT be trusted. You could say you're ANYONE.**
-- a "CA" key should never be on an Internet-connected host (if a real CA key gets compromised, chaos results)
-- what about revoking no-longer-trustworthy certs?

Those (and other) issues notwithstanding, these certs will work (at least for testing/demonstration purposes).

Enabling SSL: edit httpd.conf

In conf/httpd.conf, make sure you've uncommented:

```
Include conf/extra/httpd-ssl.conf
```

Now edit conf/extra/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")

"Can I Really Safely Dump Weak & Medium Ciphers?"

- Yes. However, if you do try it and run into some unexpected issue, backing that choice out is trivial, so go ahead and live on the cryptographic wild side! :-;
- By the way, some may wonder how we came to deploy weak ciphers in the first place. Were we just brain dead? No. In the bad old days, weak crypto was mandated for export applications by the U.S. government.* As a result, some international users only had access to crypto libraries using weak 40 bit or 56 bit ciphers. If you only offered stronger ciphers on your secure web server, in the bad old days, users with crippled web browsers couldn't connect. These days, all browsers support strong crypto, so **dump 40 & 56 bit ciphers!**
- The other factor that formerly drove some sites to use weak(er) ciphers was the computational load that use of stronger ciphers might impose. With current CPU horsepower (processor speed and core count), CPU impact has effectively become a non-issue for all but the most heavily loaded sites (and you should upgrade anyhow!)

----

* en.wikipedia.org/wiki/Export_of_cryptography_in_the_United_States

"What About That Other Parameter You Highlighted? Is There Anything Better Than TLSv1?"

- 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) 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, but in at least 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)

Browser Exploit Against SSL/TLS Tool (BEAST)

- The technical media has been all atwitter recently about BEAST, a browser-based attack exploiting long-known (heretofore theoretical) vulnerabilities that exist in widely deployed and routinely used versions of SSL/TLS.
- Unfortunately, the community still hasn't really converged around a practically workable solution to this vulnerability.
- One of the nicest summaries I've seen of what browser vendors are thinking about is: "Browsers Tackle the 'BEAST' Web Security Problem," September 29th, 2011, http://news.cnet.com/8301-27080_3-20113530-245/browsers-tackle-the-beast-web-security-problem/ (or try http://tinyurl.com/beast-summary if you prefer).
- For now, I think the best advice I can give you on this one is to continue to monitor this vulnerability.

Getting Back to Apache... Let's Start Apache With mod_ssl and Check for Any Errors

Start (or restart) Apache:

```
# /opt/local/apache2/bin/apachectl start        (or restart)
```

Check to see if there are httpd's running:

```
# ps auxw | grep httpd
```

If there aren't, check the log files for errors:

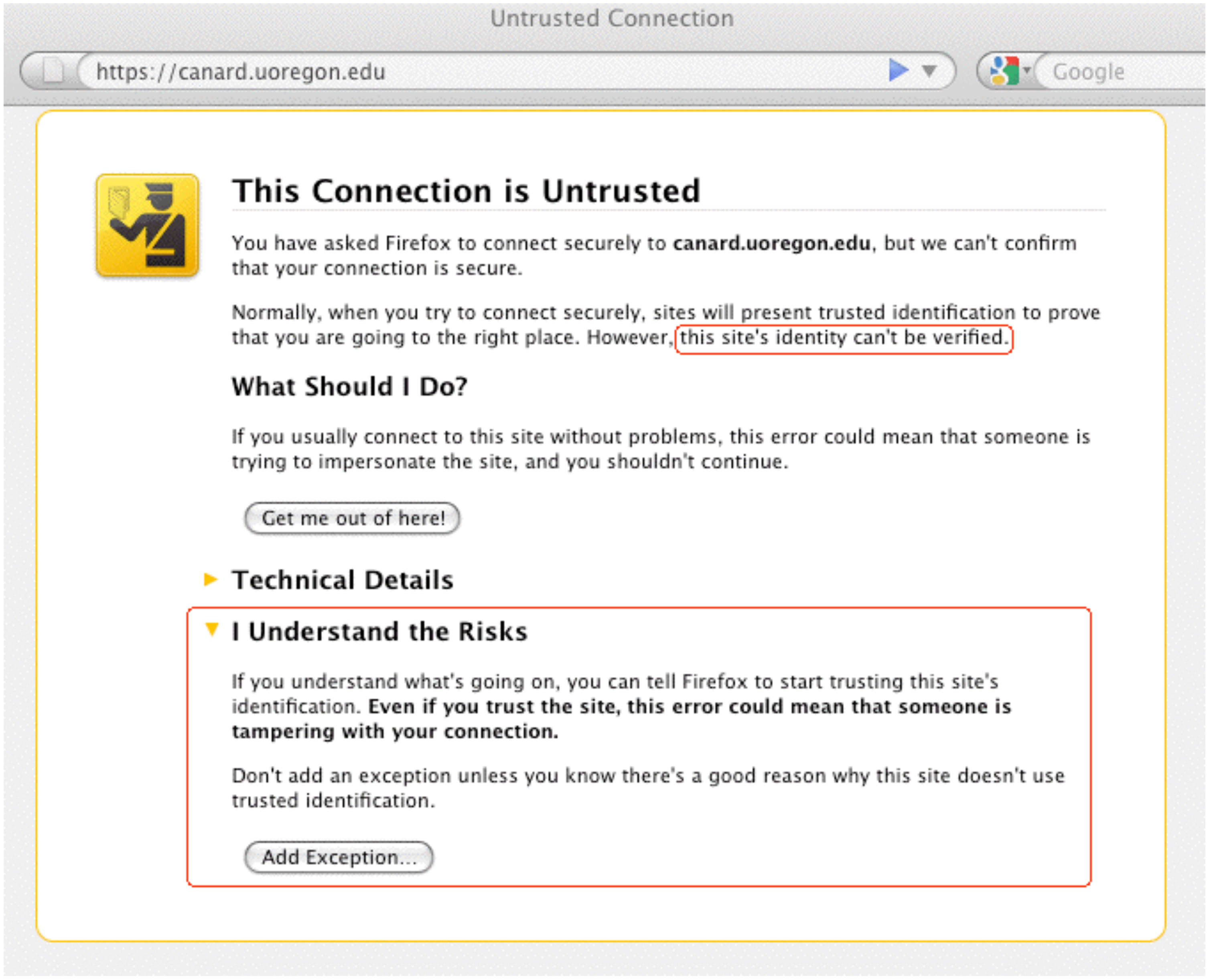
```
# tail -f /var/log/system.log              <-- ctl-C to interrupt
# tail –f /opt/local/apache2/logs/error_log
```

Everything looking okay? Now try connecting from a browser:

https://someserver.example.edu/ <-- Note the s in **https**

What will you (hopefully) see?

This Example Warning Is NOT An "Error"



This Connection is Untrusted

You have asked Firefox to connect securely to **canard.uoregon.edu**, but we can't confirm that your connection is secure.

Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.

What Should I Do?

If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.

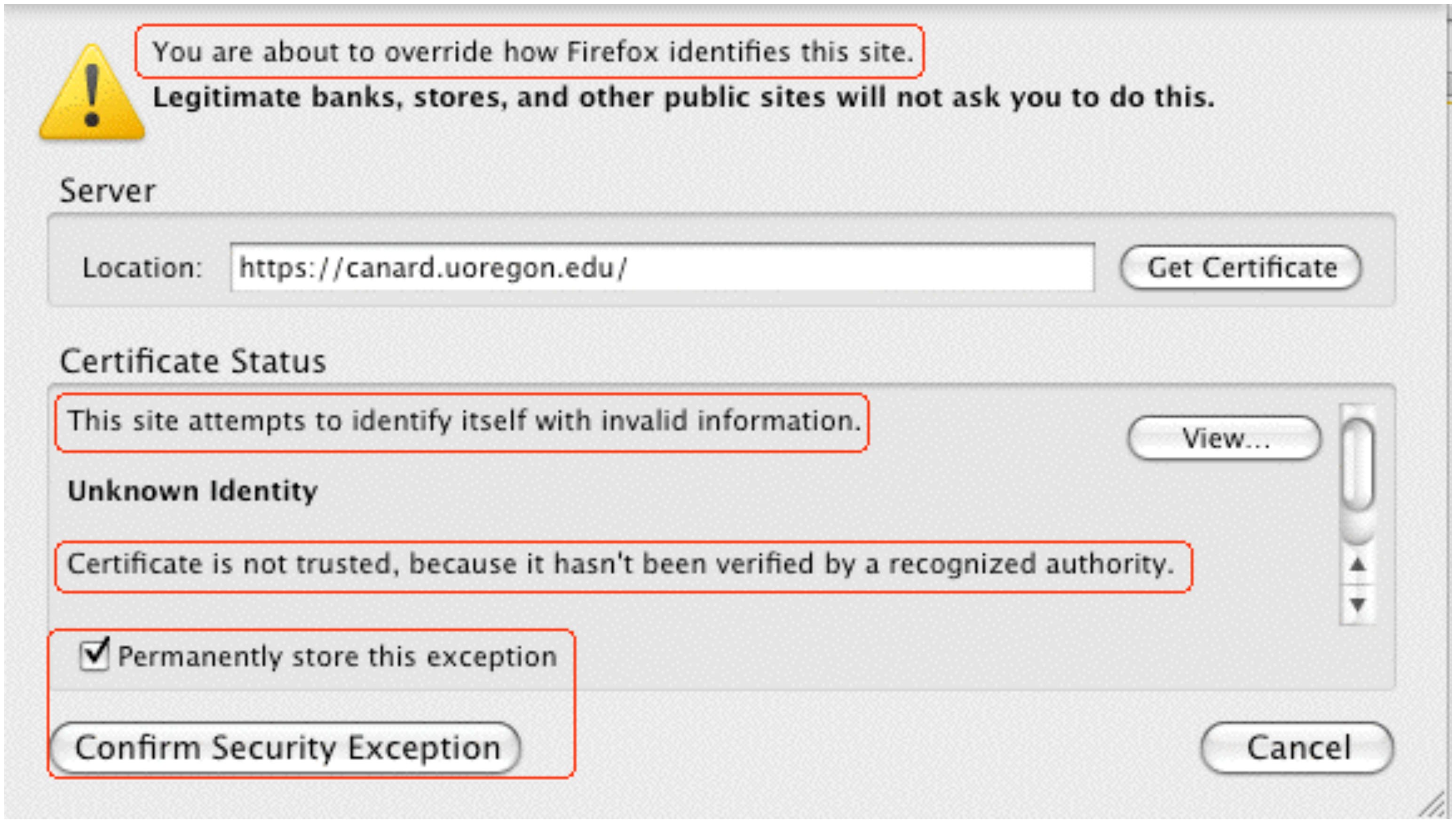
▶ Technical Details

▼ I Understand the Risks

If you understand what's going on, you can tell Firefox to start trusting this site's identification. **Even if you trust the site, this error could mean that someone is tampering with your connection.**

Don't add an exception unless you know there's a good reason why this site doesn't use trusted identification.

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



You are about to override how Firefox identifies this site.

Legitimate banks, stores, and other public sites will not ask you to do this.

Server

Location: https://canard.uoregon.edu/

Certificate Status

This site attempts to identify itself with invalid information.

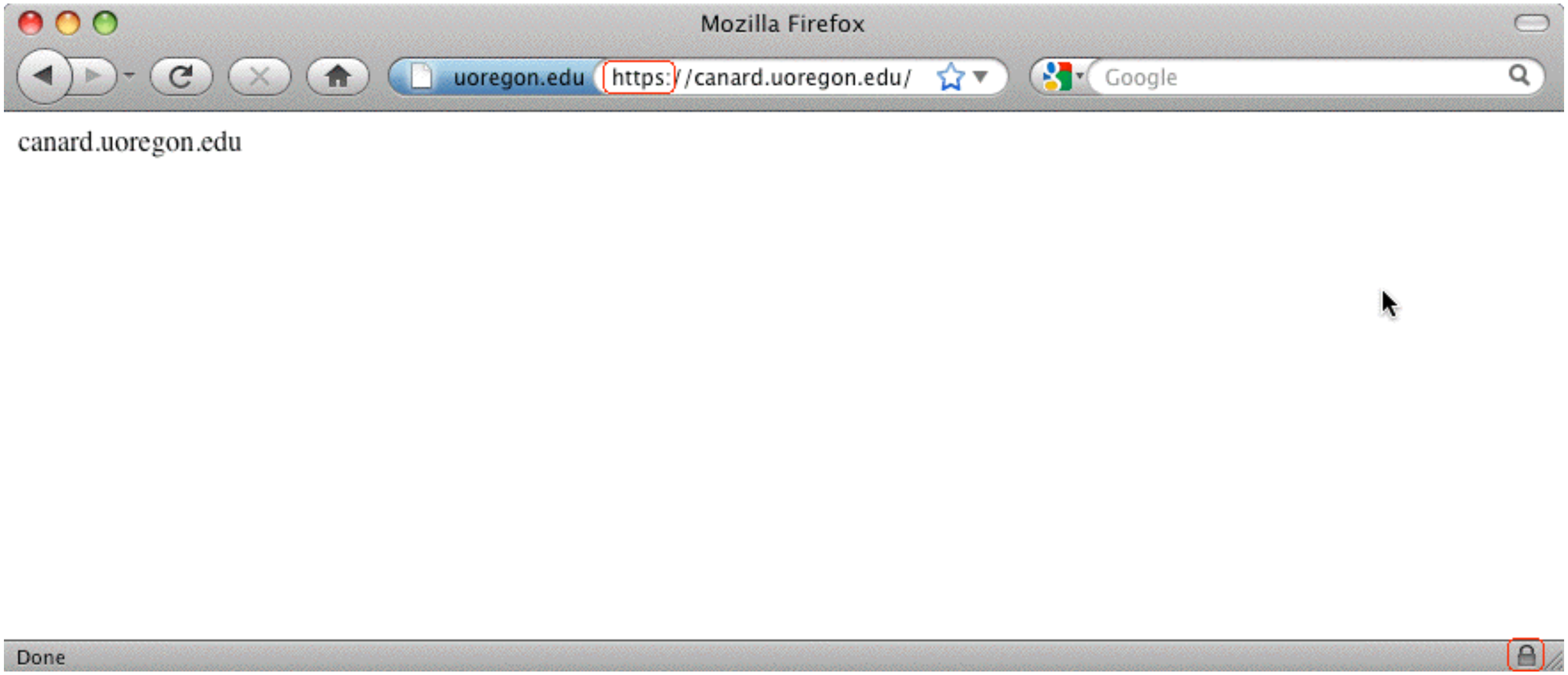
Unknown Identity

Certificate is not trusted, because it hasn't been verified by a recognized authority.

[✓] Permanently store this exception

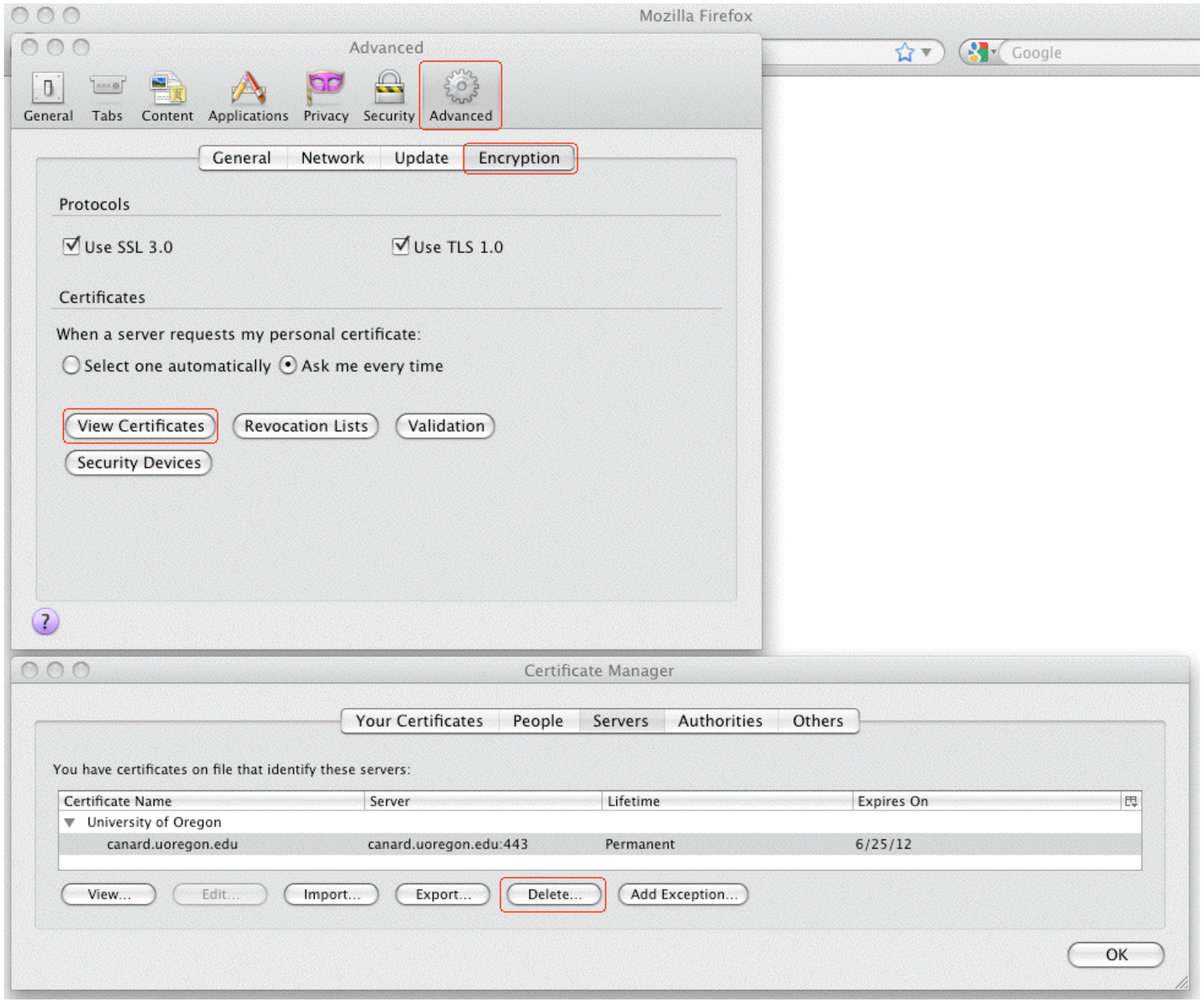
In spite of all those warnings, most users will, naturally, happily proceed to click on "Confirm Security Exception." **At that point, the SSL/TLS "trust" game is over for that server...**

Sure Looks Like A Real Trusted Site Now, Eh?



canard.uoregon.edu

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



Protocols

Use SSL 3.0 [✓]   Use TLS 1.0 [✓]

Certificates

When a server requests my personal certificate:

Select one automatically [✓ Ask me every time]

You have certificates on file that identify these servers:

Certificate Name	Server	Lifetime	Expires On
▼ University of Oregon			
canard.uoregon.edu	canard.uoregon.edu:443	Permanent	6/25/12

V. Certificate Authorities and MITM Attacks

Assume You Were Asked To Click On a URL…

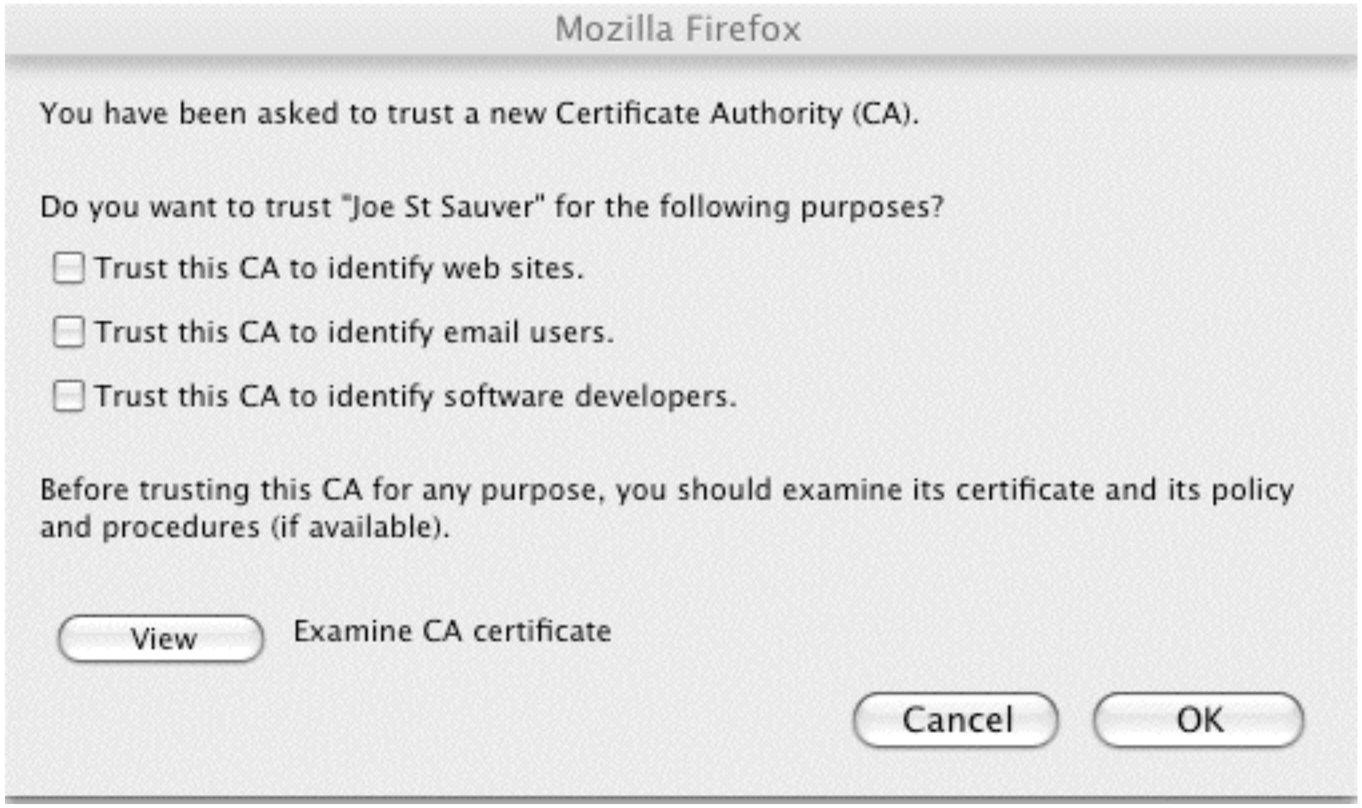
- I'm not going to give you an actual URL to click on, but let's assume that someone on the Internet asked you to click on a URL that looked something like:

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

  Would you do it? Would you click on that link? I think many people would – heck, they click on phishing URLs all the time, and malware URLs, and all sorts of stuff, 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


Mozilla Firefox

You have been asked to trust a new Certificate Authority (CA).

Do you want to trust "Joe St Sauver" for the following purposes?

- [ ] Trust this CA to identify web sites.
- [ ] Trust this CA to identify email users.
- [ ] Trust this CA to identify software developers.

Before trusting this CA for any purpose, you should examine its certificate and its policy and procedures (if available).

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 learned 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 58, 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 <u>from</u> your browser, all the way <u>to</u> 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.
- <u>They</u> (rather than the ultimate destination) can accept and decrypt your encrypted traffic. <u>They</u> 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 would be impossible 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

Just In Case I Haven't Spelled This One Out Clearly Enough: Trusting a "Random" New "CA" Is REALLY Bad

- If you decide to trust an untrustworthy "certificate authority" you may end up subsequently trusting all sorts of random sites that you shouldn't, such as sites that are **impersonating**...

  -- favorite online stores
  -- your bank, brokerage, or credit card company,
  -- your doctor's office,
  -- critical "secure" university web sites,
  -- etc., etc., etc.

- Some machines are more vulnerable to getting new random untrustworthy CAs than others...

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 malicious user could just transparently eavesdrop upon all the user's "confidential" traffic

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 <u>Really</u> Be Trusting All Default CAs?

- Commercial CAs routinely get attacked, and recently the Dutch CA DigiNotar B.V. was compromised. That hacker/cracker issued a variety of wildcard certificates, plus certs for critical and/or very high profile sites.
- As a result of that incident, all known mis-issued DigiNotar certs have been revoked. DigiNotar's root certificates have also been eliminated from the default list of trusted CAs in popular browsers and operating systems. For more, see:

  -- "DigiNotar Damage Disclosure," https://blog.torproject.org/blog/diginotar-damage-disclosure
  -- "DigiNotar Public Report, Version 1," [English language] http://tinyurl.com/diginotar-report
  -- "VASCO Announces Bankruptcy Filing by DigiNotar B.V.," http://tinyurl.com/diginotar-bankruptcy

Pruning Browser Root Cert Stores?

- The DigiNotar incident has also made some parties recommend some, um, unconventional strategies, including:

  "[...] configuring the enterprise browser platform so as to reduce the number of root CAs the enterprise relies upon. First weed out those root certificates that no one recognizes. [...] Second, weed out those root certificates that are used rarely or not at all. [...] Third, for those CAs that remain, take a few moments to interact with the CAs and determine their practices with respect to RAs and their other affiliates. [continues]"

  "From the Experts: SSL Hacked!", Corporate Counsel, Law.com, Sept 28, 2011, tinyurl.com/pruning-root-certs

What's The Big Deal About Having "Lots" of Default Trusted CAs?

- Each and every default-trusted certificate authority can potentially issue a perfectly valid (looking) certificate for any domain. Those valid (looking) certificates can then be used by attackers trying to man-in-the-middle your traffic.
- If you have over a hundred and fifty CAs that you trust by default, people worry that that's "too many," and that one or more of them may in fact be insecure or untrustworthy.
- The "obvious" (if hugely difficult) solution to this problem is to remove the "obscure" or "unneeded" CAs from that default set, as the author on the preceding slide suggests.
- In reality, however, that's a task that's fraught with many problems.

Before Giving That Strategy A Try (If You Do…)

- Be sure you can restore the default trust anchors, just in case you end up removing something you shouldn't have.
- Recognize that most people have little or no basis for recognizing or assessing trust anchors for retention or potential removal decisions. You might try saying, for example, "I'm only going to keep big American CAs," but you might be surprised at how many commonly used/critical web sites use certs from less common overseas CAs.
- If you bump into a site of that sort after you've pruned the trust anchor that would normally validate it, you (or your users!) will then need to exercise your own best judgment: is this a cert I want to permit, or not? Absent extensive personal investigation, mistakes will inevitably be made, both when it comes to accepting and rejecting certs that you're shown.

If You Feel You *Must* Prune Trust Anchors

- One strategy that *might* work would be to compare the trust anchors recognized by major operating systems and applications, keeping only those that are common to all members of that reference set. Put another way, if ALL common operating systems and browsers trust a particular CA, you might decide you might as well do so, too.
- However, if you do that, what's your plan for keeping that set of local trust anchors current over time? Is trust anchor maintenance really your favorite hobby?
- You also need to figure out what you're going to do if a trust anchor that you're nervous about has intermediate certs that are cross-certified by a trust anchor you do like... this may be more complex than you might think!
- My recommendation? PLEASE **resist** the urge to manually tweak the default operating system/browser trust anchors!

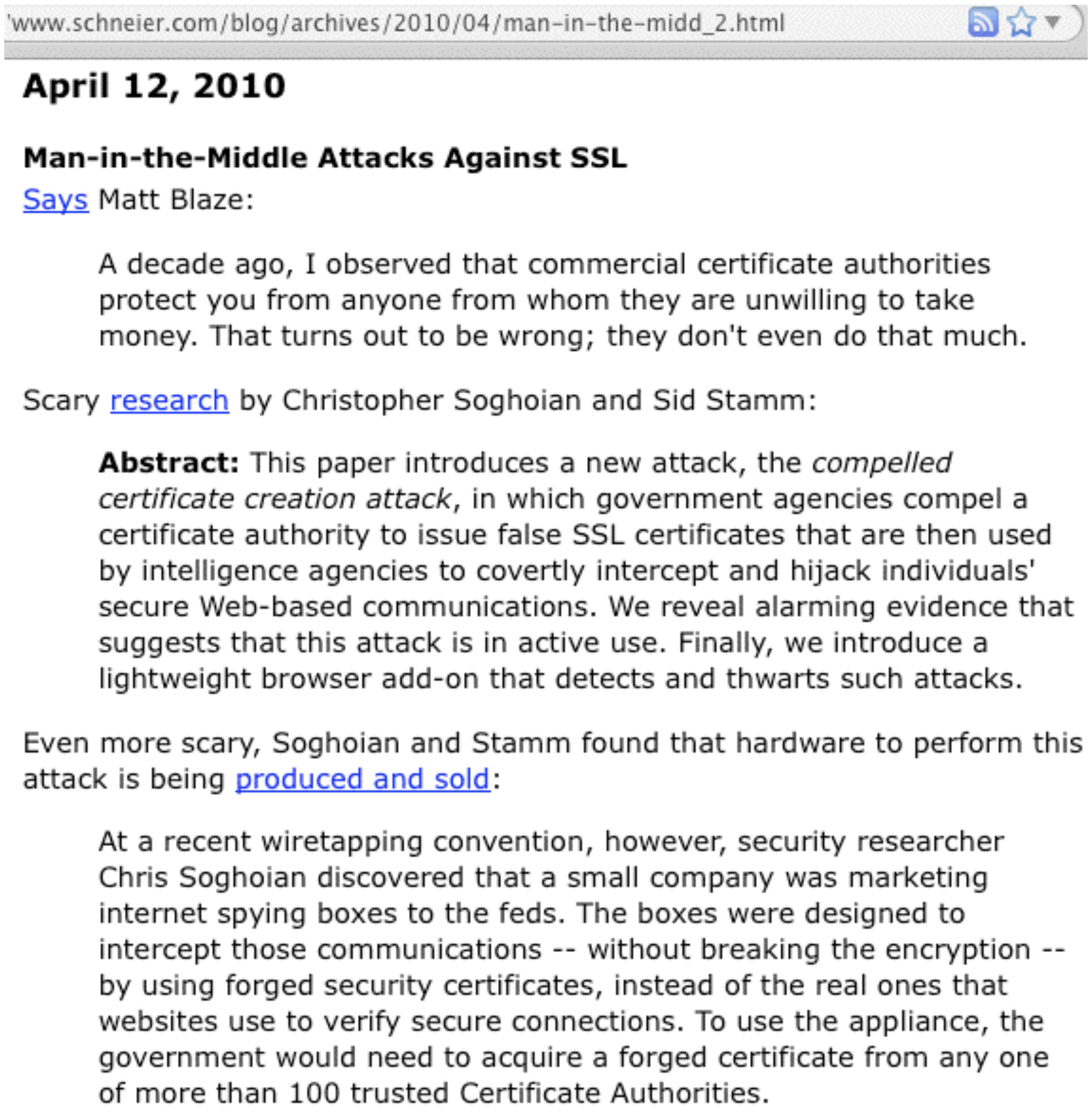
Certificate Stapling

- As mentioned on the preceding slides, currently any CA you trust can issue a seemingly valid certificate on behalf of any domain. Wouldn't it be swell if sites could specify that their site will always and only use certs from one vendor, and that any cert that might be seen from some other vendor should NEVER be trusted for their site?
- *The Good News?* This is precisely one of the use cases described by the DANE effort in the IETF. See Section 3.1 of http://tools.ietf.org/html/draft-ietf-dane-use-cases-05
- *The Bad News?* The DANE work relies on deployment of DNSSEC (which is only beginning in many parts of the net).
- At the risk of asking you to check and potentially work on Yet Another Thing, how *IS* deployment of DNSSEC coming at your campus? (Yes, this stuff really does all interlock nicely, doesn't it?)

Another Risk: Compelled Certificate Creation Attacks

- 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 <u>our</u> 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 <u>foreign</u> 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.

VI. "So Is THAT *<u>All</u>* You've Got For Us, Joe?"

No, But I've Got Limited Time And I'm Already Running Long, Aren't I? :-;

Certificate Management Processes

- One (of many) things we really haven't talked about is certificate management.

  -- WHO can order certs at your site?
  -- HOW do your certificates and keys get backed up?
  -- WHEN do certs get updated/replaced?

- All too often, you will run into certificates that either have expired, or are on the verge of expiring.

- You may really want to consider taking a more systematic approach when it comes to certificate management. The InCommon Certificate Manager includes features designed specifically to help with that, including warnings about certificates that will expire soon.

Your School (Probably) Uses More Than One Domain

- I'd be willing to *bet* that your schools all use more than one domain. You may have one dot edu domain you primarily use, but I bet you also have a hodge podge of dot coms or dot orgs or legacy edu domains in use by some parts of your school. How do those certs for all 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.), may be able to request a domain validation certificate from some CAs, and in many cases they may have already done so without even asking your permission to do 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's 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)

Certificate Validity and Revocation

- One of the most subtle and important certificate-related topics is handling certificate validation and revocation.

- Has your campus devoted any attention to making sure that user browsers "do the right thing" when it comes to checking OCSP (online certificate status protocol) and CRLs (certificate revocation lists)?

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)
- We'll talk more about this in my cert session tomorrow.

Client ("Personal") Certs

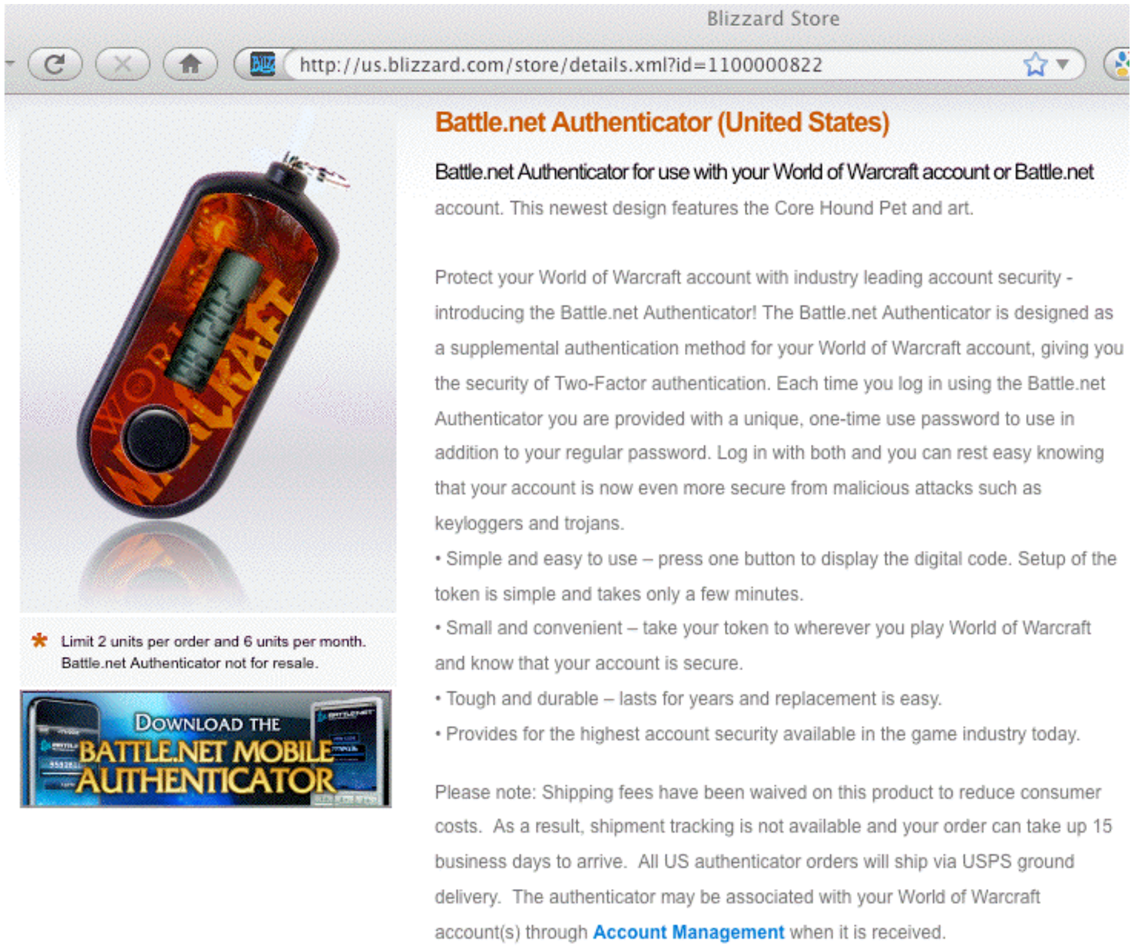
- We also don't really have much time left today to talk about client certificates, but that's another potential "game changer" from the InCommon Certificate program.
- Client certificates have had limited deployment to date in the U.S., except in the government (where the "CAC" or "PIV" is ubiquitous), but that **may** be about to change.
- Sometimes trying a new technology like client certs is just a matter of getting your "feet wet."
- For example, what about using client certs for S/MIME secured email?
- If you're interested in hearing more about client certs, please check out my other session, tomorrow. We'll explain how you can get going with them for S/MIME and other uses.

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 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?
- 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...


Blizzard Store
http://us.blizzard.com/store/details.xml?id=1100000822
Download the Battle.net Mobile Authenticator

Battle.net Authenticator (United States)

Battle.net Authenticator for use with your World of Warcraft account or Battle.net account. This newest design features the Core Hound Pet and art.

Protect your World of Warcraft account with industry leading account security - introducing the Battle.net Authenticator! The Battle.net Authenticator is designed as a supplemental authentication method for your World of Warcraft account, giving you the security of Two-Factor authentication. Each time you log in using the Battle.net Authenticator you are provided with a unique, one-time use password to use in addition to your regular password. Log in with both and you can rest easy knowing that your account is now even more secure from malicious attacks such as keyloggers and trojans.

• Simple and easy to use – press one button to display the digital code. Setup of the token is simple and takes only a few minutes.

• Small and convenient – take your token to wherever you play World of Warcraft and know that your account is secure.

• Tough and durable – lasts for years and replacement is easy.

• Provides for the highest account security available in the game industry today.

Please note: Shipping fees have been waived on this product to reduce consumer costs. As a result, shipment tracking is not available and your order can take up 15 business days to arrive. All US authenticator orders will ship via USPS ground delivery. The authenticator may be associated with your World of Warcraft account(s) through **Account Management** when it is received.

* Limit 2 units per order and 6 units per month. Battle.net Authenticator not for resale.

Hardware Tokens And Smartcards Are Another Topic…

- … that we'll talk about in tomorrow's session. :-)

- Please also don't miss Jim Jokl's PKI Subcommittee BoF on Wednesday – remember that we should be talking about the recent two factor authentication survey, client certificate installer requirements and other important topics. We'd love to have you attend if you're interested – it runs from Noon to 1PM.

Thanks for the Chance To Talk Today!

Are there any questions?