# SSL As Designed, SSL As Deployed, SSL As It Should Be

Joe St Sauver, Ph.D.
(joe@uoregon.edu or joe@internet2.edu)
InCommon Certificate Program Manager and
Internet2 Nationwide Security Programs Manager

University of Idaho Computer Security Awareness Symposium
UI Commons Crest-Horizon Room, 10:30-Noon, Oct 13th, 2011

http://pages.uoregon.edu/joe/designed-deployed-should-be/

# 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, all under contract through UO.

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

- If you're hoping for a "sales pitch," you're going to be disappointed, because that's not my goal today. (If you would like to know about the InCommon Certificate Program, check out http://www.incommon.org/cert/ ).
I may mention it once or twice, but that's about it.

# 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 today's audience is diverse, 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 most of this talk approachable for everyone.

- Anyhow, our first questions are, and probably must be, **"Why are we particularly interested in <u>web site security</u>?"** and **"Why focus on that 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 entrust 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



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 Does Now Have 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.

# 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*
  -- 2006: TLS 1.1 (aka SSL 3.2)        *supported" version,*
  -- 2008: TLS 1.2 (aka SSL 3.3)        *believe it or not!*

\* 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 read 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 *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 conceal 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, let's take a couple of minutes to review how to install/update Apache

# The Higher Ed SSLlab Score Distribution for 119 Dot Edus

| score | Frequency | Percent | Cumulative Frequency | Cumulative Percent | |
|---|---|---|---|---|---|
| **0** (score<20) | 7 | 5.88 | 7 | 5.88 | <-- **F** |
| **48** (score>=20) | 10 | 8.40 | 17 | 14.29 | <-- **D** |
| **52** (score>=50) | 29 | 24.37 | 46 | 38.66 | <-- **C** |
| **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** (score>=65) | 8 | 6.72 | 79 | 66.39 | <-- **B** |
| **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 | |

# 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 may be 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 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 early October 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]

# "Do We Really <u>NEED</u> To Upgrade?"

- **Yes.** Apache releases often address security issues which, if left unpatched, can result in your server potentially being exploited or (more commonly) DDoS'd.

- If you want to see the "gory details" for what's been patched in Apache 2.2.x, check out (for example): http://httpd.apache.org/security/vulnerabilities_22.html

- And if you think that the script kiddies don't actually have production quality attack code leveraging these vulnerabilities, see for example http://www.metasploit.com/modules/ (grep that page for "Apache Range header" to see one specific example of an Apache-focused exploit module)

# 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 many (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 Firewall (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

Note: 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

**Note: 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, another thing 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 <u>IS</u> 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

# 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 <u>lot</u> 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 <u>Really</u> 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 with Apache? 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 <u>are</u> 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?**

# <u>This</u> Example Warning Is NOT An "Error"

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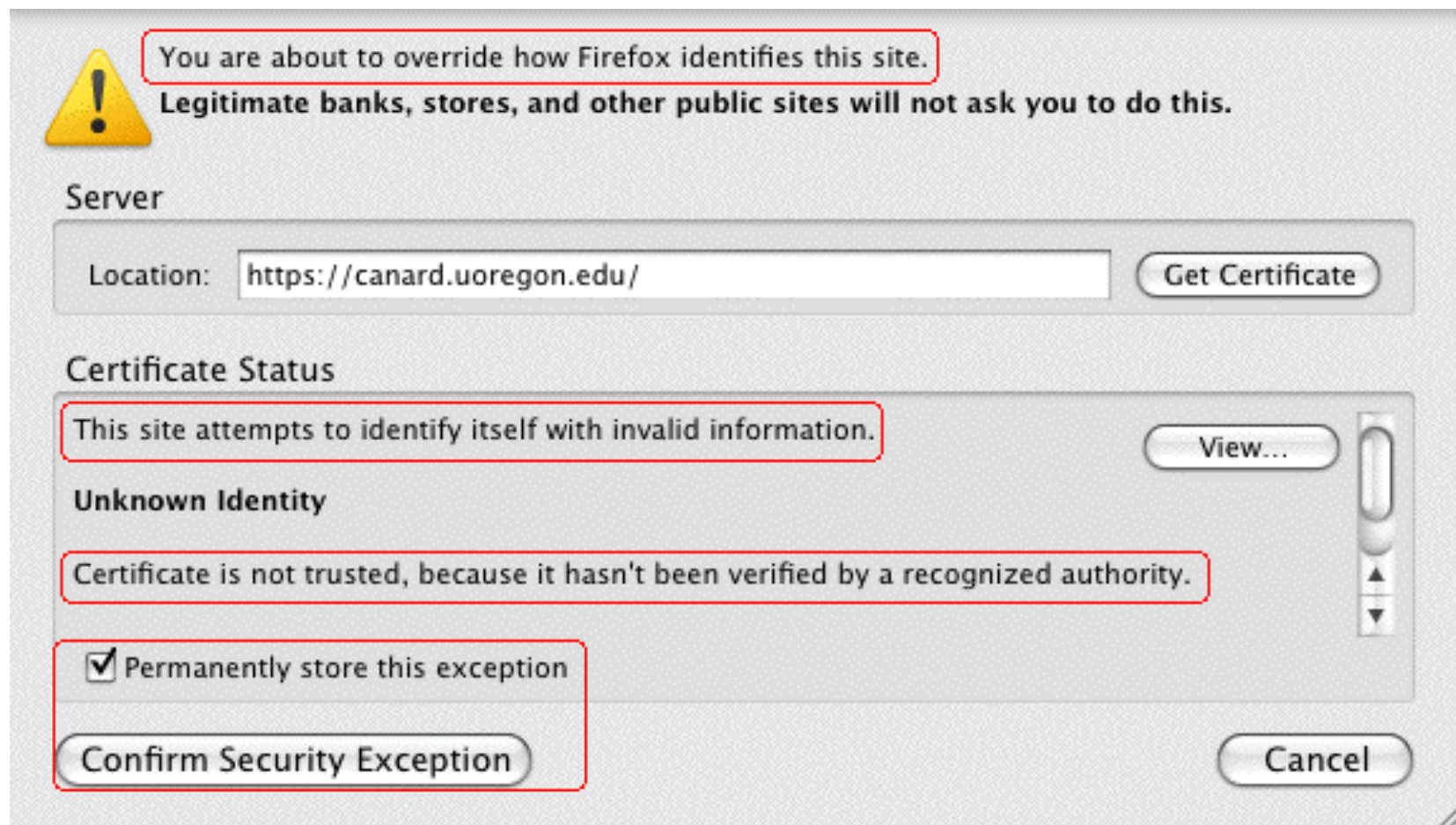


In spite of all those warnings, most users will, naturally, happily proceed to click on "Confirm Security Exception." **At that point, the SSL/TLS "trust" game is over for that server...**

# Sure Looks Like A Real Trusted Site Now, Eh?

# What If You Wanted To Delete A (Mistakenly) Trusted SSL/TLS Server Certificate? In Firefox Preferences...
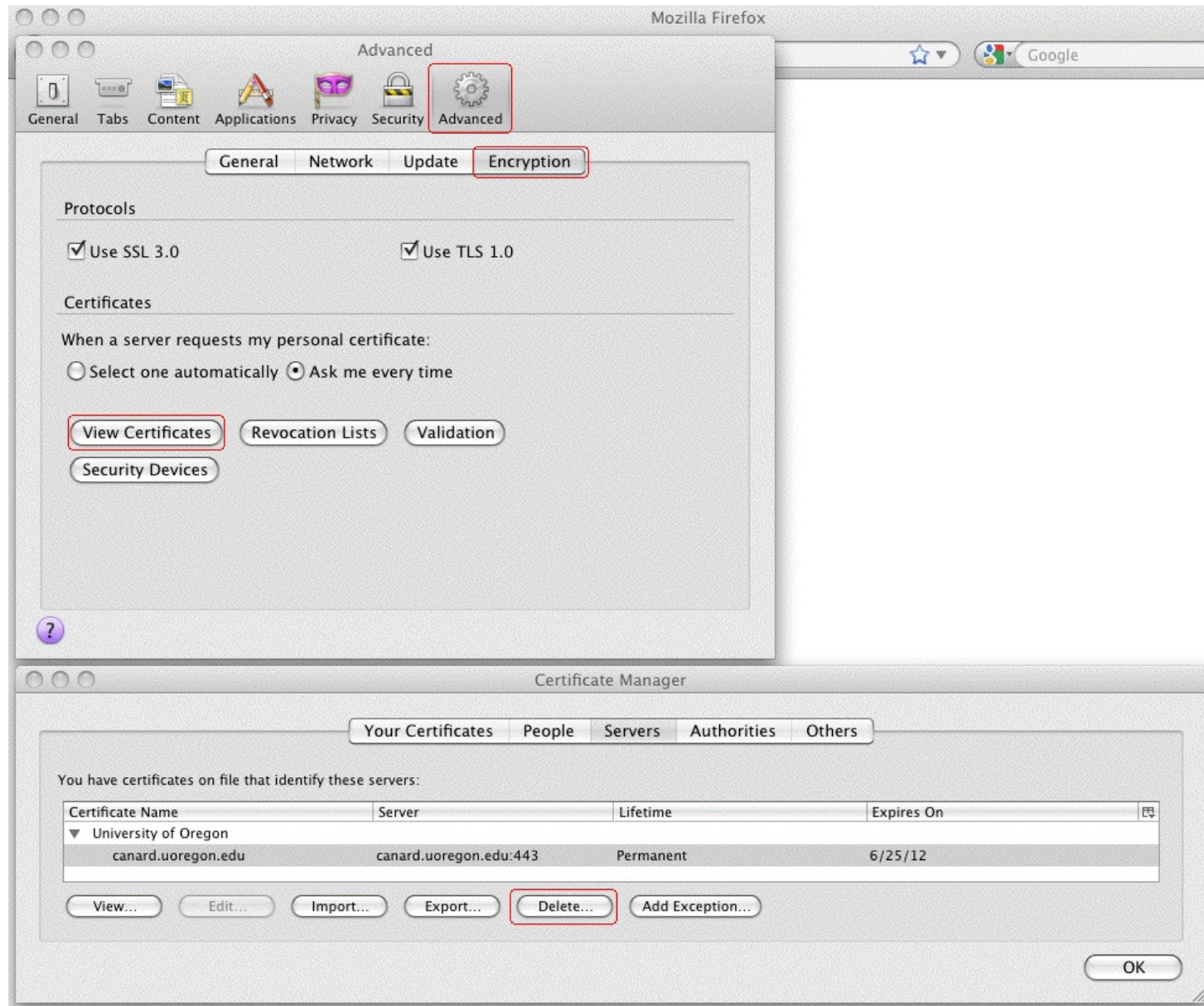
# V. Certificate Authorities and MITM Attacks

# 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).

( View )    Examine CA certificate

( Cancel )    ( OK )

**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 55, 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

# "Could Someone *Really* MITM Me?"

- **Yes. MANY** MITM approaches exist. Just to mention a few:

- Advertise one or more "evil twin" wireless access points, targeting local wireless sites that aren't doing 802.1X

- Attack local **layer 2** switching infrastructure using ARP poisoning with something like Cain and Abel
  ( www.oxid.it/cain.html )

- Attack wide area routing by injecting more specific routes (see "Revealed: The Internet's Biggest Security Hole," www.wired.com/threatlevel/2008/08/revealed-the-in/ )

- Attack **DNS** mapping of fully qualified domain names to IP addresses (via DNS cache poisoning attacks, or "DNS changer" malware), see for example www.metasploit.com/modules/auxiliary/spoof/dns/bailiwicked_domain

- Institutionally install an application layer gateway doing DPI on all traffic (hey, their network, their rules, right?)

# 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 <u>Very</u> 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 <u>All</u> 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 secure web traffic w/o being detected.

- 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.

# <u>Before</u> Giving That Strategy A Try (If You Do...)

- Be <u>sure</u> 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 <u>only</u> those that are common to <u>all</u> 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 <u>really</u> your favorite passtime?

- 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

## 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.

76

# 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 originally made these slides during the summer of 2011.

# VI. Certificate Management

# WHO Can Order Certs At Your Site?

- Typically, whether you intend for this to be true or not, anyone who can monitor any one of a number of standard role accounts (commonly this list is admin@domain, administrator@domain, postmaster@domain, hostmaster@domain, webmaster@domain), or anyone who is listed in whois as a domain's tech or admin point of contact for that domain, can get a "domain-validated" cert from one or more certificate authorities.

- Like "make your own change" day at the store, this may be popular with users, but not a practice that is particularly security affirming, eh?

- I'd recommend implementing policies that firmly lodge certificate procurement and distribution with a relatively small cadre of individuals, not your postmaster, webmaster, etc., but perhaps part of your security team.

# Your School (Probably) Uses More Than One Domain

- I'd be willing to *bet* that your schools all use more than one domain. For example, you may have one dot edu domain you primarily use, and if you do your own DNS you may have a pretty good idea of what servers exist in that domian.

- However, I bet you <u>also</u> have a hodge podge of dot coms or dot orgs or legacy dot edu domains in use by at least some parts of your school.

- How do those certs for all THOSE domains get handled? ["Buy your own certs day!"]

- Is that how you expected this to all work? :-;

# 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.

- Using user-generated called calls as a warning mechanism that one of your certs needs to be replaced is *not* a recommended and highly professional practice.

- Take advantage of scheduling tools to set up reminders for when your certs are getting moderately close to expiring. (Ninety days in advance might be a nice "early warning," and you should have a fallback notification at the thirty day mark, just in case things "fall through the cracks").

- You might even want to get into a routine of replacing your current certs with new certs every summer or winter vacation as part of your scheduled preventive maintenance during periods of low usage (much in the way you might replace the batteries in your home smoke detectors).

# What Duration Certificates Should You Buy?

- Assume you _can_ buy certs with durations running from one year to three years or even longer in some cases. (If your CA is in the default Mozilla set of root trust anchors, Mozilla requires revalidation of information included in SSL certs at least every 39 months, see https://wiki.mozilla.org/CA:Problematic_Practices )

- WHAT duration _should_ you buy?

- The temptation will probably be to just buy the longest duration cert you can get, thereby obtaining any multiyear discount that you might be able to earn while also minimizing ordering and installation hassles, but from a security point of view, maybe that's less than ideal.

- Hypothetically, perhaps domain registration validity periods should put an outer bound on certificate validity periods? (Should a one year domain reg have a three year cert?)

# Wildcard Certificates

- While you can buy an individual certificate for each host in your domain, you can also buy multi-domain certs for multiple hostnames, or even buy "wildcard" certs.

- For example, you could get a cert for server1.example.edu, server2.example.edu, server3.example.edu, and server4.example.edu or even a cert for *.example.edu

- Having one cert for all your hosts certainly streamlines your certificate ordering process (and could also be cheap)

- However, as that cert gets passed out to sysadmin after sysadmin for installation around campus, how secure will it be? If any ONE of those hosts gets broken into, ALL hosts will need to replace their now-compromised certificates.

- Wildcard certs also don't do a very tight job of binding a server's identity to the credentials its using.

# Signature Types/Lengths

- Certs are signed by certificate authorities. In the past, some CAs may have used MD5-based signatures, or relatively short 1024 bit signatures.

- These <u>must</u> be phased out/replaced. See https://wiki.mozilla.org/CA:MD5and1024 and www.incommonfederation.org/cert/doc/2048-bit-Certificates.pdf

- **Note:** this will most often come up as an issue for cert <u>renewals</u>, where someone has long had a 1024 bit-signed certificate, only to learn that it will not be able to be renewed. If someone has had a 1024 bit-signed cert it will need to be replaced with a 2048 bit cert (which will require creation of a new key pair and a new CSR).

# Server Gated Cryptography Certs

- Another legacy of the "bad old days" is "Step Up" or "Server Gated Cryptography" (SGC) certs. SGC certs were designed to accommodate users connecting from cryptographically-crippled "export grade" browsers. Those bad old days are long gone, yet some sites **are** still using SGC certs (I wouldn't be).

- If users <u>are</u> continuing to use antique browsers that rely on the availability of SGC, those browsers are inherently insecure and those users should be encouraged to update their browsers to something at least moderately current.

- A nice article on this topic is at 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 users' browsers "do the right thing" when it comes to checking OCSP (online certificate status protocol, RFC2560) and CRLs (certificate revocation lists, RFC5280)? OCSP and CRLs are <u>supposed</u> to be used to signal the revocation status of certs to browsers and other apps.

- Unfortunately, some browsers don't check OCSP/CRLs!

- If revocation checking isn't done, users risk trusting a revoked certificate, which is generally a pretty bad idea.

- Because support for OCSP and CRLs may vary in browsers, I recommend encouraging users to use a current version of Firefox, since it does a good job of managing use of OCSP and CRLs by default.

# VII. "Tell Me About Extended Validation Certs!"

# A Sample Secure Web Page
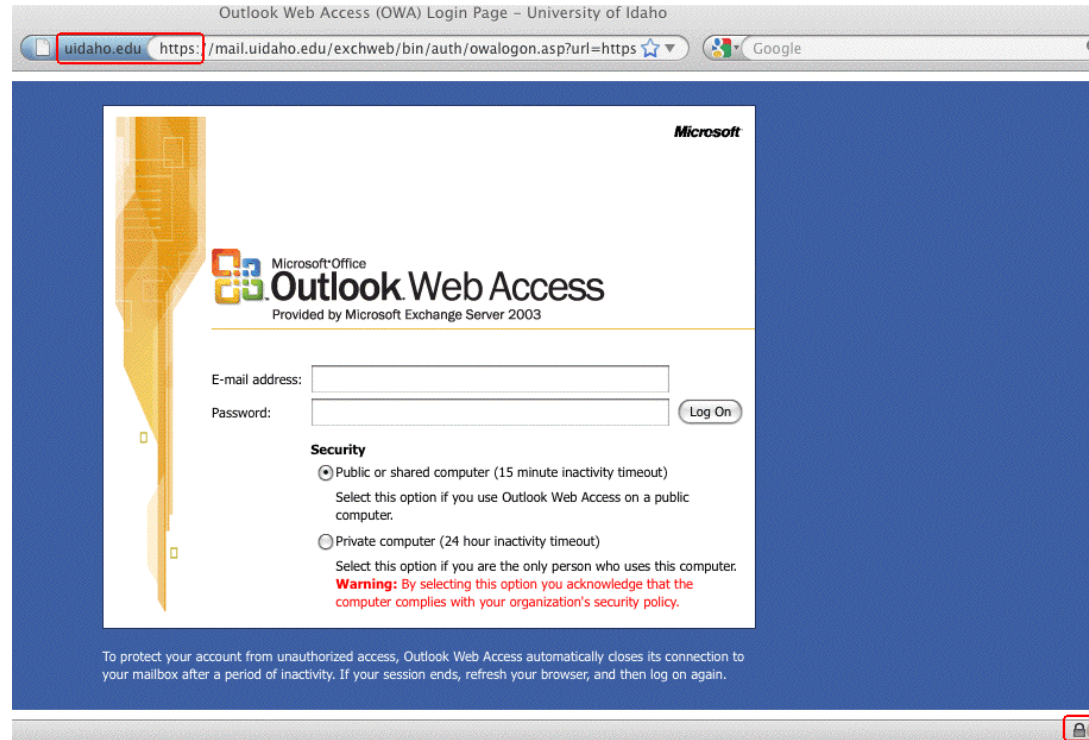


- In Firefox, a variety of user interface elements are meant to "cue" or help the user to notice that this is a special secure web page, and not just a "regular" one. Notice:
  -- the blue field with the domain in the address bar
  -- the https URL prefix (with an "s", standing for "secure")
  -- the little padlock on the bottom margin of the window

# Those Cues are Subtle, Can Be Overlooked or Faked, And May Confuse Some Users

- Many folks won't even notice that the "s" on https, or the presence of that little padlock, when visiting a secure page.

- Sometimes, phishers attempt to trick users by adding a "padlock" favicon.ico ( http://en.wikipedia.org/wiki/Favicon ) to a web site they're abusing, hoping that users will think that the padlock icon they're seeing means that this is a secure site, even though it isn't.

- The prominent blue-colored area near the web address is hard to overlook, but will users understand what that color is meant to connote?

- What if the user <u>was</u> curious, and wanted more information? For example, what if users started clicking on some of those user interface cues, looking for more information?

# More Information *IS* Potentially Available...

- While looking at that web site in Firefox, users can click on the blue area to the left of the web address, asking to see "More Information" and then asking to "View Certificate:"



General | Details

This certificate has been verified for the following uses:

SSL Server Certificate

**Issued To**

Common Name (CN)          mail.uidaho.edu
Organization (O)          mail.uidaho.edu
Organizational Unit (OU)  Domain Control Validated
Serial Number             01:00:00:00:00:01:28:89:9A:8C:E8

**Issued By**

Common Name (CN)          GlobalSign Domain Validation CA
Organization (O)          GlobalSign nv-sa
Organizational Unit (OU)  Domain Validation CA

**Validity**

Issued On                 5/11/10
Expires On                5/11/12

**Fingerprints**

SHA1 Fingerprint          56:25:48:28:3A:FB:2C:35:AB:0C:F8:86:99:BB:C1:25:B2:8A:00:2B
MD5 Fingerprint           78:64:F7:8C:7C:2B:99:64:FF:7D:A0:0A:0B:83:C9:05

Set Nickname...

Close

# But Users Usually <u>Won't</u> Bother Looking at Cert Info

- After all, why should they? Secure web sites "work" even if users doesn't look at the certs, eh?

- Also, certificate-related information panes prominently feature all sorts of obscure/intimidating/cryptic numbers and even basic vocabulary that gets used in unexpected or confusing ways (for example, consider the terms "Common Name" or "Fingerprint" as shown on the preceding slide)

- Even if a user did look at that information, but then had questions, where would they go to get those questions resolved? Secure sites typically are pretty much a "take it or leave it" proposition, right? You really only have two choices: use it, or don't use it.

# Users May Have Little Choice But to Accept

- A recent study by Holz, Braun, Kammenhuber and Carle* found that less than 1-in-5 of the certificates they observed had both a correct host name and a valid certificate chain, thereby allowing the cert to be determined to be cryptographically valid.

- The rest of the time, one of the following is true:
  1) appropriate certificate checks aren't occurring, 2) users are proceeding notwithstanding substantial cert problems, or 3) an awful lot of sites using certs aren't accessible!

- In truth, the problem may even be worse than that – not only do users <u>accept</u> certs they shouldn't, they routinely draw <u>unsound inferences</u> from a site's use of an SSL cert...

  ----

  * Preprint shared on the Randombit Cryptography mailing list 29 Sep 2011, see
  http://lists.randombit.net/pipermail/cryptography/2011-

# The Default Mental Thought User Process

- Many users have been implicitly or explicitly **mis-trained**, and now -- almost as an article of faith -- many seem to **incorrectly** believe that:

  "As long as you see https in the address bar... or you see that little padlock down on their web browser's bottom margin... or you see the blue colored field up near the web address... Then you can proceed to 'safely' enter passwords, credit cards, etc., there."

- That is, of course, completely **crazy.**

- Peter Gutmann of the University of Auckland has some classic examples of "interesting" sites that have valid certs (and some real "mainstream" sites that have invalid ones) in "PKI as Part of An Integrated Risk Management Strategy for Web Security," EuroPKI 2011, http://www.cs.auckland.ac.nz/~pgut001/pubs/pki_risk.pdf

# What Certificates Can (and Can't) Do

- Remember, securing web sites through use of cryptographic certificates was meant to accomplish three things:
  -- protect your information from eavesdropping
  -- protect your information from tampering
  -- reassure you that you're dealing with the site you really wanted to deal with, and not a fake/imposter site

- That's *NOT* what users think they're getting from a site secured with a certificate.

- **Users have a far simpler (wrong!) notion:**
  *SSL "Secured" Site? --> The Site Must Be <u>Trustworthy</u> (fair, honest, able to be relied on, etc)*

- SSL certs are actually like the services of a notary public. A notary public certifies that she saw you sign, and that your government ID matches your name and signature, NOT that the contract she saw signed was worthwhile one.

# Put Simply: Certs Are NOT About "Reputation" or "Trust"

- **Good** people and organizations can get certificates. Really **bad** people and organizations (including criminals!) can **also** get (at least some types of) certificates (see Gutmann's talk, mentioned previously in this section).

- It would be, and is, a critical/terrible mistake to assume that just because a web site has a valid SSL certificate, that that site is trustworthy! Many users do NOT "get" this, so this is a point worth stressing if you talk with them about certs

- Technically, certs themselves don't even provide protection against eavesdropping or tampering, that's actually done by the cryptographic key pair that's behind or underneath the certificate.

# Certs Bind *Identities* (Of One Sort or Another) To *Cryptographic Key Pairs*

- What are those "cryptographic key pairs" we just mentioned?
  Behind every certificate there lives:
  -- a public key that can be freely shared with anyone, and
  -- a corresponding (secret!) private key.

- The public key forms the foundation for each Certificate Signing Request (CSR). The CSR gets forwarded to the certificate authority (CA), which then **validates the identity of the requesting entity** (one way or another), issuing a certificate that cryptographically signs the public key, binding the requesting entity's identity to it.

- What varies from cert type to cert type is the type and thoroughness of the validation process that gets employed. That validation can vary from extensive (in the case of EV certs) to nothing at all (self-signed certs).

# Four Levels of Identity "Validation"

**1) "Self-Signed" certs:** these certificates **haven't been validated** by a broadly recognized certificate authority. You have no assurance whatsoever that those cryptographic credentials really belong to who you think they belong to. Maybe they do, maybe they don't. You simply don't know.

**2) Domain Validation (DV) certs:** an automated email with a unique code gets sent to an email address associated with the domain for which a cert has been requested. This email
goes to a standard role account (such as postmaster) or a point of contact address listed in the domain's whois. The certificate requesting party then follows the instructions in that email ("click on this link") to demonstrate "control" over that **domain name.** DV certs are probably the most common type of SSL certificate. DV certs simply bind a certificate to the applicant's domain name.

97

# Four Levels of "Validation" (continued)

**3) Organizational Validation (OV) certs:** OV certs, such as the ones that InCommon issues via Comodo, require a more careful verification of the applicant's business identity and applicant roles. Use of an OV cert will typically not result in any prominent indicator or mark that signifies that status in most browsers, although if you check the certificate manually, you will normally see the organization name (and *not* just a domain name) actually listed in the cert.

**4) Extended Validation (EV) certs:** Extended validation certs are the least common sort of cert. Before an EV cert gets issued, a more thorough investigation of the identity of the applicant gets conducted per the requirement developed by the Certificate Authority and Browser forum. Sites that use an EV cert **will** display the organization's name in a "green bar" in current generation web browsers.

# EV Cert Indications Vary
# From Browser To Browser
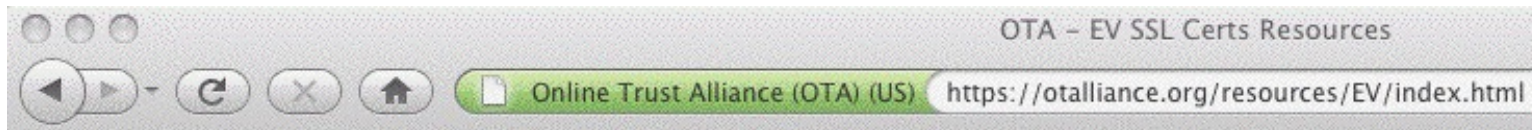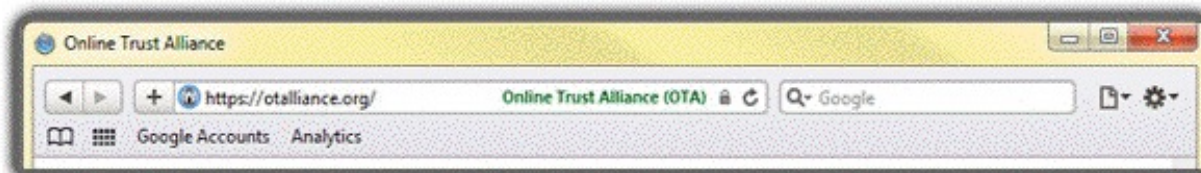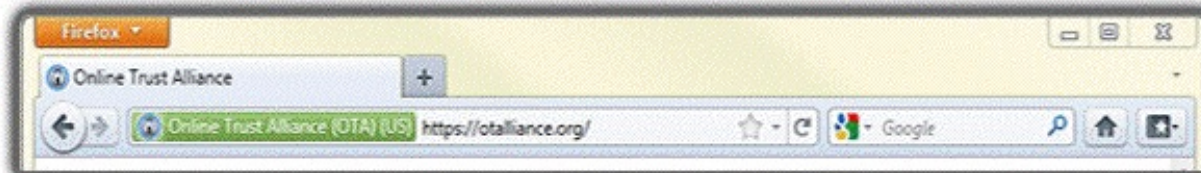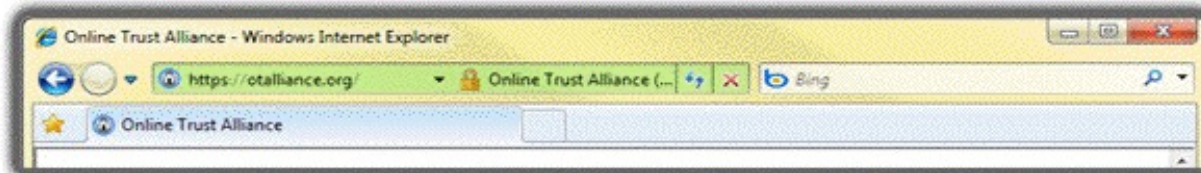
# Choice of A Cert Validation Type

- Why do some sites use self-signed certs, while others use DV certs, OV certs, or EV certs? Three factors normally drive site selection of one or the other: cost, convenience and consumer confidence.

- **Cost:** self-signed certs are free. DV certs are quite cheap. OV certs are usually more expensive (since they involve manual processing/validation of applicant details). EV certs, involving the most thorough review, are the most expensive.

- **Convenience:** You could make your own self-signed certs. DV certs, since they are normally processed on a wholly automated basis, can be issued in near real-time. OV certs normally take longer (since they involve manual processing). EV certs take the most time (and paperwork!) of all.

- **Consumer Confidence:** Hopefully this increases from self-signed certs to DV certs to OV certs to EV certs.

# EV's Role: Reversing A "Race To The Bottom"

- For a long time there weren't all these types of certs. In the old days, there were only self signed certs, and carefully verified commercial certs.

- Then certificate authorities began to compete on price. If you're competing on price and have very thin profit margins, you can't afford to do extensive validation of each applicant and still make a profit. You need to substitute automation -- and many CAs did. They began to verify that you had control over a *domain name*, rather than that you controlled a *business entity*. Importantly, in many ways, these cheap DV certs had little that tangibly distinguished them from higher quality/higher cost OV certs.

- **Extended validation certificates were created in an effort to claw back consumer confidence, particularly for banks and other prime phishing targets. BUT...**

# Netcraft survey indicates slow adoption of Extended Validation SSL certificates

By Dancho Danchev | April 26, 2011, 4:05am PDT

**Summary:** *According to the latest Netcraft SSL Survey, Extended Validation SSL certificates still only account for 2.3% of all valid third party certificates analyzed by the company.*

Extended Validation (EV) SSL certificates
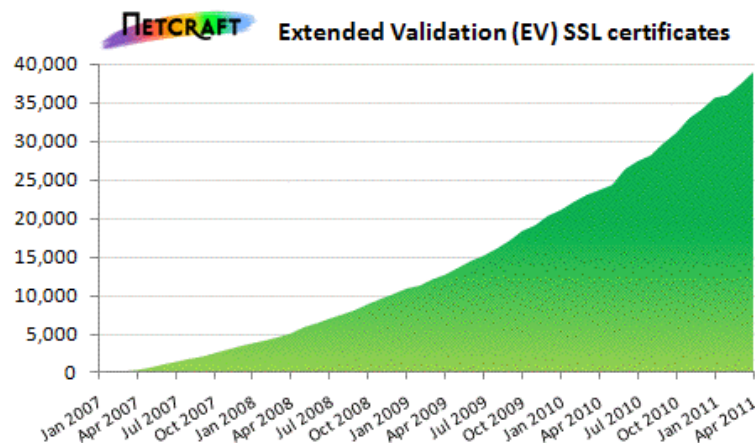
According to the latest Netcraft SSL Survey, **Extended Validation SSL certificates** still only account for 2.3% of all valid third party certificates analyzed by the company. The steady, but slow adoption is attributed to both, pricing and site verification concerns. The survey finds that extended validation SSL certificates are most prevalent on high traffic or financial web sites, and are used to further establish a trusted relationship between the web site and the visitor.

Restricting the survey's sample to the busiest 1,000 websites in the world, 81 sites accepted HTTPS connections and presented a valid SSL certificate. Nearly a third of these certificates used Extended Validation – a far higher proportion than the 2.3% share of all certificates.

# Why The Slow EV Rollout? Issue One: Cost

- While EV certs do a nice job of potentially improving confidence in critical sites, extended validation certificates are still quite uncommon because they have traditionally been **expensive** to obtain (hundreds of dollars per cert), and many sites simply couldn't afford to obtain them.

- Some good news:

  Cost has ceased to be an impediment to deploying EV certificates, at least for sites participating in the InCommon Certificate Program, because extended validation certificates are included in the InCommon Certificate Program at no additional charge.

# EV Issue Two: Paperwork

- Even though the InCommon Certificate program can make cost a non-issue for Cert Service subscribers, getting extended validation certs WILL still require your site to do some paperwork, including typically producing a "lawyer letter."

- Details of the restrictions associated with EV certs, and what's required in terms of paperwork can be seen here: https://www.incommon.org/cert/evcerts.html

- I wouldn't let the potential paperwork deter you from applying for an EV cert – it really isn't THAT bad (and besides, those of us in academia excel in processing paperwork, right :-) )

# EV Issue Three: User Awareness

- The final issue that has inhibited EV cert adoption has been a lack of user education and awareness.

- Even with a change in color on the browser address bar, many users don't "get" what that color implies, or why extended validation certificates are better than a regular SSL certificate.

- Nonetheless, despite those issues, we ARE seeing higher education sites deploying EV certificates.

- A few examples of universities that are using EV certs follow...

# A University Site That Uses an EV Cert

# A Second Example Using An EV Cert

# And A Third Example Using an EV Cert

# And A Final Example Using an EV Cert

# Is YOUR School Using EV Certs? Should You Be?

- If you have a security-critical web site that collects/uses:

  -- Passwords or cookies for authentication or "login"
  -- Personally identifiable information (such SSNs)
  -- Financially sensitive information (such as credit card info)
  -- Medical information (e.g., HIPAA-covered information)
  -- Grades or other FERPA-covered student records

  Or, if you have sites that may be a priority target for spoofing, such as wireless authentication or VPN sites, then yes, I think you SHOULD be using Extended Validation ("green bar") certificates for those sites.

- **"If it's a site that matters, GO GREEN!"**

# Coming Back to EV Issue 3 For A Second: Have You Trained Your Users About EV Certs?

- It is probably unrealistic to expect users to know about and understand extended validation certificates on their own – you should consider an awareness program to help make your users aware of the role and implications of extended validation certificates when they encounter them.

- You should particularly train them that if they customarily see a green bar cert for a critical site, but then one day they suddenly don't -- STOP and find out what's going on! Did they perhaps make a typo or otherwise accidentally go to the wrong site? Or, is something sinister happening?

- EV certs have one other advantage: Criminals like to work from the shadows and hide their identity so law enforcement can't track them down and arrest them. It's hard for a criminal to hide their ID AND obtain an EV cert.

# Are EV Certs <u>Cryptographically</u> "Stronger?"

- No. EV certificates, just like self-signed certs, EV certs, or OV certs, all have 2048 bit signatures these days. They are not <u>cryptographically</u> stronger.

- EV certs ARE <u>procedurally</u> stronger when it comes to establishing who's "behind" those certificates.

- EV certs are also <u>less common</u>, which helps to reduce the chance that a look-alike site will have a "green bar" cert the way a real site might.

# VIII. HTTP Strict Transport Security

# Certificates: They're Not Just For <u>Critical</u> Web Content Anymore

- For a long time, most sites only deployed certs for critical content, leaving the vast majority of routine web traffic flowing over the network unencrypted.

- Why? The usual answers (valid or not) were all or some of:
  -- we do encrypt <u>login</u> info, that's the only real worry...
  -- users don't care; why bother?
  -- why buy expensive certs when they aren't needed?
  -- it's a hassle obtaining, installing and maintaining certs
  -- we don't want to have to accept the "performance hit" that comes with doing all that crypto!
  -- debugging problems will be harder with encryption
  -- encrypted traffic can't be cached or proxied
  -- incoming encrypted traffic can't be scanned for malware
  -- I'm too busy/I'll do it "real soon now"

# Then, The World Encountered Firesheep...

- If you're not familiar with Firesheep, see http://codebutler.com/firesheep (24 October 2010)

- Firesheep is an application that does a nice job of demonstrating that encrypting <u>just</u> the user's login session is not enough (at least if a web site relies on cookies for authentication and access control): even if an attacker couldn't capture your username and password, they could still capture an unencrypted cookie, and after that, the attacker would then have full control of your account.

- This wasn't a new vulnerability, but creation of Firesheep made it apparent to everyone that this was a practical (rather than theoretical) worry.

- The only real solutions? Don't rely on cookies to carry critical security info -- **or** <u>encrypt **everything** with https.</u>

# HTTP Strict Transport Security (HSTS)

- What we need is a way for sites to declare that ALL traffic for their domain MUST be sent via https, and ONLY https.

- If we just wanted to ENCOURAGE use of https on a site, a formal protocol isn't absolutely necessary. Any site could simply decide to start using https to secure the web pages on their site, and voila, it could be done. However, it's easy for a user to accidentally request a page via http (instead of https), or for a web programmer to mistakenly link to an unencrypted local web page rather than an encrypted one.

- Fortunately, HSTS provides a way to say that a site MUST use https only. See: "HTTP Strict Transport Security (HSTS)," Hodges (Paypal), Jackson (CMU) & Barth (Google), tools.ietf.org/html/draft-ietf-websec-strict-transport-sec-02 (expires February 6th, 2012)

# Enabling HSTS

- Enabling HSTS on a web site that uses Apache is pretty easily done, see the description at "Adding HTTP Strict Transport Security (HSTS) to Apache Virtual Hosts," http://linux.dashexamples.com/2011/08/adding-http-strict-transport-security-hsts-to-apache-virtual-host/

- The one required additional HSTS header <u>must</u> be sent via an https: page – that header will be ignored if it is sent via an unencrypted http page. As a result of that requirement, and also due to limited browser support, OWASP emphasizes use of a 301 permanent redirect instead: www.owasp.org/index.php/HTTP_Strict_Transport_Security (note that while approach works with any browser, it doesn't directly preclude use of self-signed certs or some other corner cases that HSTS explicitly addresses)

# Browser Support for HSTS

- To be candid about one disappointing point: browser support for HSTS is not currently really where I'd like it to be, and that's a shame, because browsers play a key role in recognizing and enforcing use of the HSTS protocol.

- *The good news?* HSTS *is* at least currently enabled in recent versions of Firefox and Chrome (e.g., see for example http://www.chromium.org/sts ). Another helpful point: even if you use a browser that doesn't support HSTS, that browser's non-support of HSTS won't actively break anything.

- *The bad news?* There <u>are</u> major/important browsers that don't currently have support for HSTS: Internet Explorer, Safari, and Opera do <u>not</u> have support for HSTS at this time. (Likewise, I don't believe that HSTS has made it into many mobile device web browsers). Talk to your vendors!

# Name-Based Virtual Hosting and https Usage

- One other consideration: when it comes to regular (non-https) hosting, many sites use <u>name-based</u> virtual hosting. In name-based virtual hosting, dozens or even hundreds of domains may get hosted on a single shared IP (e.g., see httpd.apache.org/docs/2.2/vhosts/name-based.html )

- <u>Traditionally</u>, secure web sites needed <u>IP-based</u> hosting, with each secure web site residing on a dedicated address. If you have lots of secure web sites, doing IP-based hosting could rapidly deplete your pool of available addresses.

- Server Name Indication ("SNI") eliminates that requirment if you're running a current secure web server and browser. See http://wiki.apache.org/httpd/NameBasedSSLVHostsWithSNI

- *Caution:* Some browsers on some operating systems may not have support for SNI.

# Mixed Scripting and Mixed Display

- While you're tightening things up and promoting use of https everywhere, you may particularly want to note the problem of "mixed scripting," where an https page loads a script, cascading style sheet or plugin resource over an insecure (http, instead of https page).

- Also bad: when an https page loads an image, iframe or font over http, a related if somewhat less serious problem that's sometimes called "mixed display".

- A nice summary posting on this issue is available at "Trying to End Mixed Scripting Vulnerabilities," June 16, 2011, http://googleonlinesecurity.blogspot.com/2011/06/trying-to-end-mixed-scripting.html (the comments to that post bring up some interesting examples of prominent sites that apparently continue to have issues in this regard)
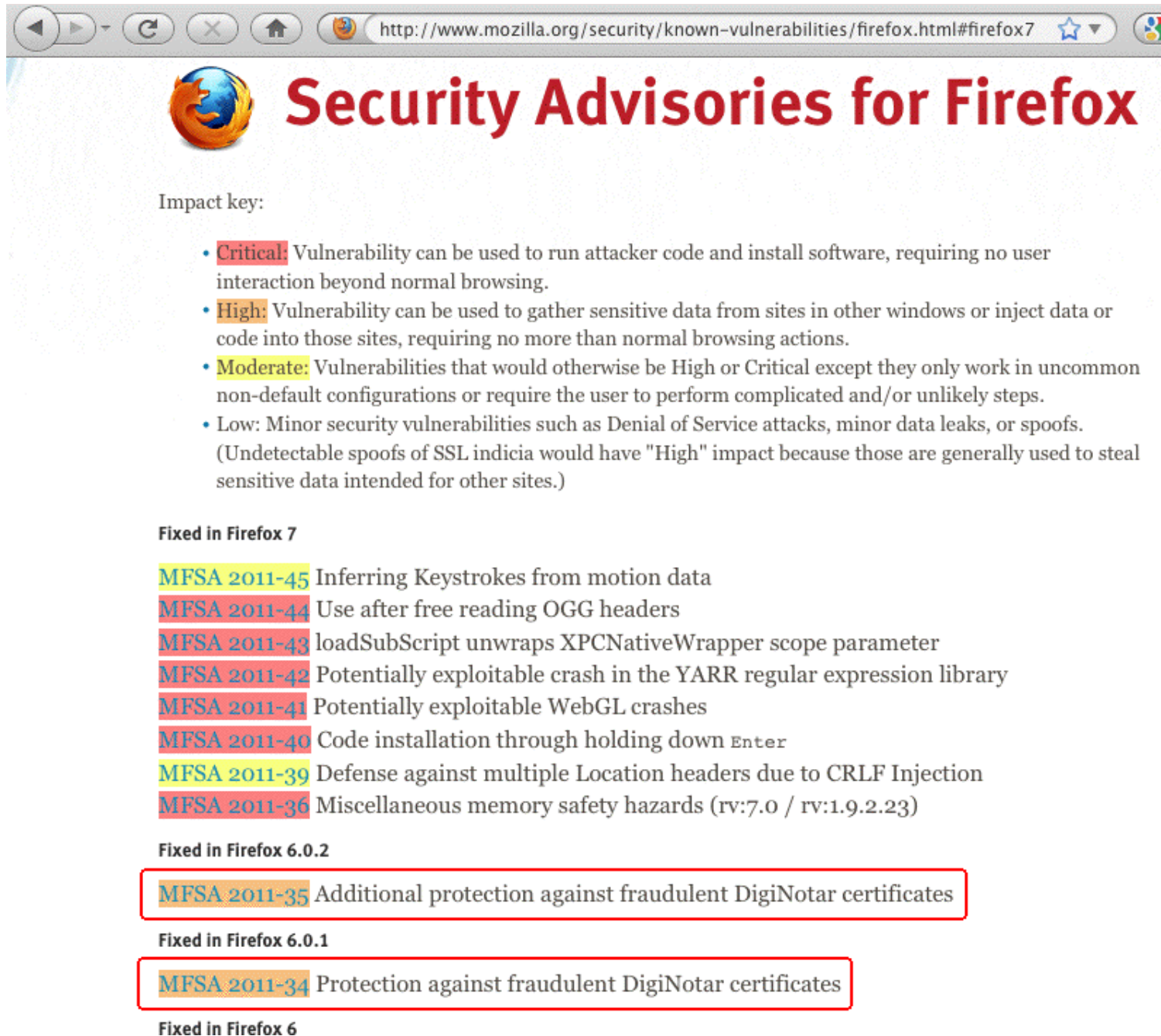
# Quick "Take Aways" For HSTS

- **The old default:** unencrypted http for most web pages, with SSL/TLS security only where it's "absolutely needed"

- **The new default:** plan on using encryption (https) everywhere.

- **Things to check:**
  -- Can my web server software support SNI? If not, do I have enough IPv4 address space to do IP-based hosts? Should I be requesting more?
  -- Are we recommending a browser that supports SNI (as well as HSTS)? Are there any old legacy Windows XP user desktops that we might need to get updated?
  -- If we moved to HSTS, would we encounter mixed scripting or mixed display issues?

# IX. Speaking of Browsers...

# We've Been Largely Talking About What We Should Be Doing On The Server Side, But What About Your Users' Browsers?

- Some recommendations won't exactly be surprising.

- For example, when it comes to dealing with secure sites, just as with pretty much anything else, browsers need to be kept patched up-to-date, including any browser plugins. Encourage use of Secunia PSI/CSI/OSI (see secunia.com) or at least http://www.mozilla.org/en-US/plugincheck/

- You may wonder what cert-related stuff needs patching or updating in the browser itself. Well, Browsers include lists of trusted root certificate authorities, and they also include hardcoded blacklists of certificates which are known to be <u>untrustworthy</u>. When a DigiNotar-type incident occurs, browsers need to update those lists. See, for example, Firefox's list of recent security updates...

# Firefox's List of Recent Security Issues



**Security Advisories for Firefox**

Impact key:

- Critical: Vulnerability can be used to run attacker code and install software, requiring no user interaction beyond normal browsing.
- High: Vulnerability can be used to gather sensitive data from sites in other windows or inject data or code into those sites, requiring no more than normal browsing actions.
- Moderate: Vulnerabilities that would otherwise be High or Critical except they only work in uncommon non-default configurations or require the user to perform complicated and/or unlikely steps.
- Low: Minor security vulnerabilities such as Denial of Service attacks, minor data leaks, or spoofs. (Undetectable spoofs of SSL indicia would have "High" impact because those are generally used to steal sensitive data intended for other sites.)

**Fixed in Firefox 7**

MFSA 2011-45 Inferring Keystrokes from motion data
MFSA 2011-44 Use after free reading OGG headers
MFSA 2011-43 loadSubScript unwraps XPCNativeWrapper scope parameter
MFSA 2011-42 Potentially exploitable crash in the YARR regular expression library
MFSA 2011-41 Potentially exploitable WebGL crashes
MFSA 2011-40 Code installation through holding down Enter
MFSA 2011-39 Defense against multiple Location headers due to CRLF Injection
MFSA 2011-36 Miscellaneous memory safety hazards (rv:7.0 / rv:1.9.2.23)

**Fixed in Firefox 6.0.2**

MFSA 2011-35 Additional protection against fraudulent DigiNotar certificates

**Fixed in Firefox 6.0.1**

MFSA 2011-34 Protection against fraudulent DigiNotar certificates

**Fixed in Firefox 6**

124

# Example of One of Those Specific Advisories



http://www.mozilla.org/security/announce/2011/mfsa2011-35.html

## Mozilla Foundation Security Advisory 2011-35

### Additional protection against fraudulent DigiNotar certificates

| | |
|---|---|
| **Impact:** | High |
| **Announced:** | September 6, 2011 |
| **Product:** | Firefox, Thunderbird, SeaMonkey |

| | |
|---|---|
| **Fixed in:** | Firefox 6.0.2 |
| | Firefox Mobile 6.0.2 |
| | Firefox 3.6.22 |
| | Thunderbird 6.0.2 |
| | Thunderbird 3.1.14 |
| | SeaMonkey 2.3.3 |

**Description:** As more information has come to light about the attack on the DigiNotar Certificate Authority we have improved the protections added in MFSA 2011-34. The main change is to add explicit distrust to the DigiNotar root certificate and several intermediates. Removing the root as in our previous fix meant the certificates could be considered valid if cross-signed by another Certificate Authority. Importantly this list of distrusted certificates includes the "PKIOverheid" (PKIGovernment) intermediates under DigiNotar's control that did not chain to DigiNotar's root and were not previously blocked.

**References:**

- Interim Report September 5, 2011: DigiNotar Certificate Authority breach "Operation Black Tulip"
- https://bugzilla.mozilla.org/buglist.cgi?bug_id=683261,683449,683883

# What About Smart Phone/Tablet Browsers?

/www.mobilityfeeds.com/mobility-feed/2011/09/diginotar-bogus-certificates-make-android-and-apple-users

## DigiNotar bogus certificates make Android and Apple users vulnerable to privacy attacks

Eight days after the discovery that a fraudulently issued Web credential actively targeted Iranians as they accessed their Gmail accounts, millions of people who rely on Google and Apple products remain vulnerable to similar attacks.
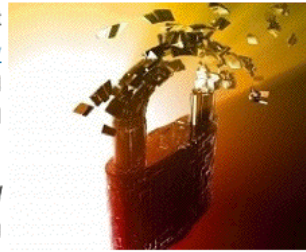
The inaction of Google in updating its *Android* Operating System (OS) and Apple in making changes to its *iOS* and *Mac OS X* is even more striking given a report issued Monday that found that a security breach on Dutch firm DigiNotar minted at least 530 additional counterfeit certificates for domains including addons.mozilla.org, Skype, and various Microsoft update sites.

While updates issued over the past week have protected users of the major browsers and Email clients, users of Google *Android*–based devices, *iPhones*, *iPads*, and Apple *Safari* on *Mac* remain susceptible unless they take special precautions.

> "*Apple is characteristically quiet again when it comes to security and at a time when its users need their help most of all*," Andrew Storms, Director of Security Operations at nCircle, wrote in an Email. "*Users are left going the unofficial route looking for experts outside of Apple to tell them how to protect themselves.*"

Apple has steadfastly declined to comment on unpatched security vulnerabilities in its products.

Developers of Google's *Chrome* browser have done a good job of communicating the risks users face from the fraudulently issued *DigiNotar* certificates. In the past week, as additional information has come to light, they have issued two updates designed to prevent the bogus credentials from being accepted by the browser when users encounter them.

Google officials have been considerably more inert when it comes to threats the certificates pose to users of *Android*, the world's most widely used smartphone OS.

A Google spokesman declined to comment for this post.

*Android* users who want to take security matters into their own hands can install the latest version of WhisperCore, a privacy app that will block most SSL certificates signed by *DigiNotar*.

# Detecting Changes in Cert Usage on Firefox

- If you routinely use ssh, you know that if/when a server changes its keys, ssh notices and warns you that something's awry. Paraphrasing: "The credentials you saw last time are not the same as the credentials you're being given this time. Watch out! Someone may be doing a man-in-the-middle attack against you!" That's potentially a very helpful alert.

- A similar process <u>doesn't</u> happen when it comes to web browsers and secure web sites. You might see one certificate today, and a <u>completely</u> different certificate tomorrow, and as long as both are validly signed, your browser won't complain.

- CertificatePatrol is a Firefox browser plugin that helps expose those sort of changes for the https websites you visit.

Google

*Certificate Patrol* 2.0.12
by Carlo v. Loesch, Gabor Adam Toth, 20after4

Your browser trusts many certification authorities and intermediate sub-authorities quietly, every time you enter an HTTPS web site. This add-on reveals when certificates are updated, so you can ensure it was a legitimate change.

**+ Add to Firefox**

★★★★☆
42 user reviews
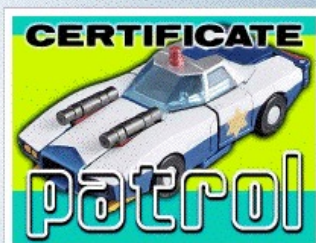16,910 users

▢ Add to collection
⮜ Share this Add-on

**Enjoy this add-on?**
The developer of this add-on asks that you help support its continued development by making a small contribution.

**♥ Contribute**
$10.00 suggested



**About this Add-on**

Your web browser trusts a lot of certification authorities and chained sub-authorities, and it does so blindly. "Subordinate or intermediate certification authorities" are a little known device: The root CAs in your browser can delegate permission to issue certificates to an unlimited amount of subordinate CAs (SCA) just by signing their certificate, not by borrowing their precious private key to them. You can even buy yourself such a CA from GeoTrust or elsewhere.

It is unclear how many intermediate certification authorities really exist, and yet each of them has "god-like power" to impersonate any https web site using a Man in the Middle (MITM) attack scenario. Researchers at Princeton are acknowledging this problem and recommending Certificate Patrol. Revealing the inner workings of X.509 to end users is still deemed too difficult, but only getting familiar with this will really help you get in control. That's why Certificate Patrol gives you insight of what is happening.
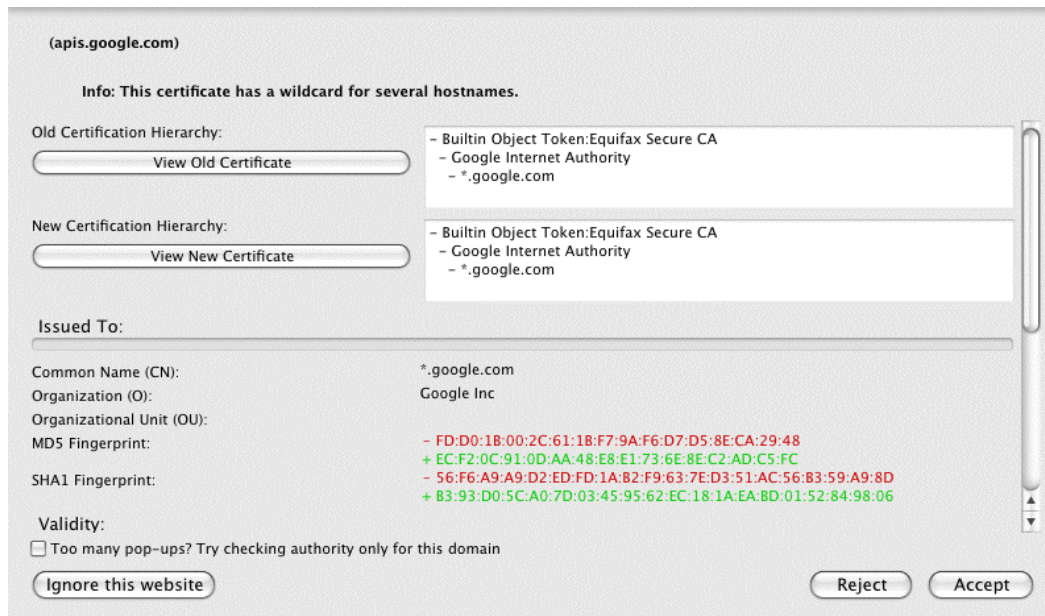
⌂ Add-on home page
⌂ Support site

Version 2.0.12 Info
September 15, 2011
Released under Mozilla Public License, version 1.1

128

An Example of
An Anomaly
Detected by
CertificatePatrol:
The Googleplex,
Temporarily
Out of Sync?

# Moxie Marlinspike's "Convergence"

- You may also hear about Convergence, a Firefox add-in that uses a network of "notaries" who collectively and anonymously tell you if what <u>you're</u> seeing for a site's certificate is consistent with what <u>they've</u> seen for a site's certificate. (see http://convergence.io/index.html )

- This sort of check is helpful if your worry is that someone may be trying to do a local "man-in-the-middle" attack against you. If they attempt to do this, the cert you'll see will differ from the cert that the rest of the world will see, and Convergence will hopefully alert you to that.

- Some in the community have expressed concerns about the Convergence model's scalability; see, for example: "Why Not Convergence?" http://www.imperialviolet.org/2011/09/07/convergence.html

- I suggest that sites just test/evaluate Convergence for now.

# 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 yet. One of the nicest summaries I've seen of what browser vendors <u>are</u> 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 issue. Currently there are LOTS of bad/inadequate answers, unfortunately.

# Thanks for the Chance To Talk Today!

Are there any questions?