Realtime publishers

The Shortcut Guide™ To Certificates in the Enterprise

sponsored by

thawte™
it's a trust thing™

Don Jones

Copyright Statement

Chapter 4: Certificate "Gotchas"—What You Don't Know *Will* Hurt You

I'm always running into administrators and users who have "given up" on certificates because they've run into troubles with them. There's no question that certificates can be a bit complicated, and there are definitely a few "gotchas" that can make them seem even *more* complicated—and even annoying. In this chapter, we'll explore some of the more common certificate snags, and look at ways of avoiding them entirely.

Self-Signed Certificates and Web Browsers

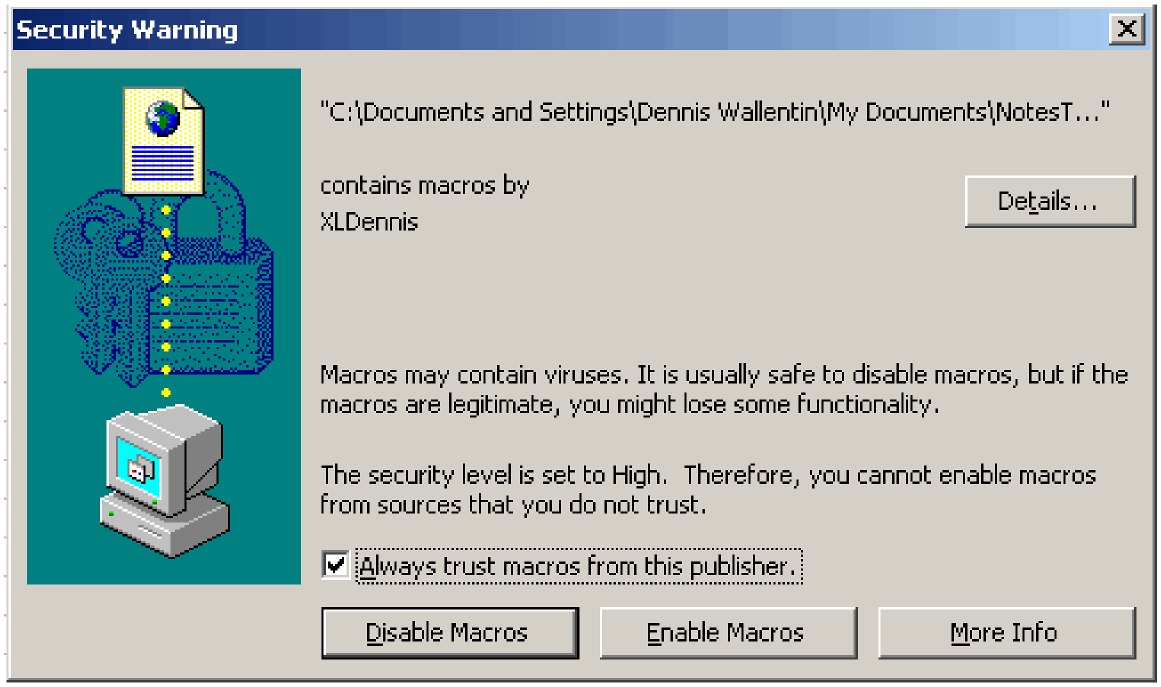
I think the biggest and most difficult-to-understand problem in the world of certificates is the *self-signed* certificate. These can come into existence in a number of different ways.

Self-Signed Local Certificates

One type of self-signed certificate is the kind a developer or user can create for themselves, on their local computer, using a tool such as Microsoft's Makecert.exe. This certificate isn't issued by an actual Certification Authority (CA); instead, it's made from scratch right on the local computer. It's a perfectly valid certificate; it can be used to encrypt email, sign code, or do any of the other things certificates can do. The problem? Nobody will trust it. This type of certificate is the digital equivalent of you scratching out a "driver's license" in crayon on a piece of construction paper. Your artwork might be top-notch, but nobody besides you is going to think it's a real driver's license.

So why bother with these at all? Convenience. By using a self-signed certificate, a developer can sign their applications in a way that *their local computers will trust.* This lets them run under-development code in an environment that *requires* digital signatures, but it doesn't let that code run anywhere else—because nobody else will trust that certificate. At some point, the developer will finish the application and sign it with a real, trusted certificate. The self-signed certificate is intended primarily as a convenience feature so that the developer doesn't need to purchase or otherwise obtain a real certificate strictly for local development.

The problem arises when a user who generates one of these self-signed certificates doesn't understand their limitations. In my experience, the most common scenario is Microsoft Office users who develop complex macros or other Office-based scripts, and then use a self-signed certificate to digitally sign their work. Everything runs fine on their computer—but when they try to share the document with another user, the certificate shows up as untrusted, as Figure 4.1 shows. Depending on the configuration of Office in that environment, the user might or might not be allowed to trust the self-signed certificate, and either way the dialog box presents confusion.


Security Warning

"C:\Documents and Settings\Dennis Wallentin\My Documents\NotesT..."

contains macros by
XLDennis

Macros may contain viruses. It is usually safe to disable macros, but if the macros are legitimate, you might lose some functionality.

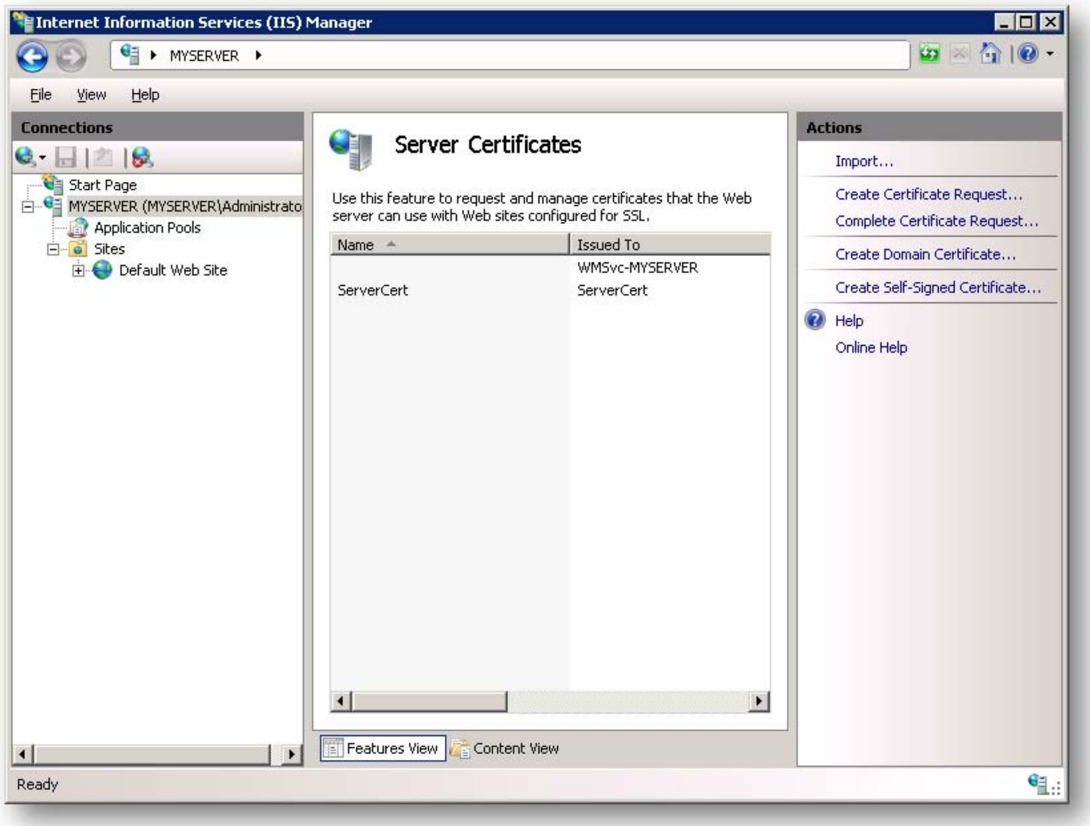
The security level is set to High. Therefore, you cannot enable macros from sources that you do not trust.

[✓] Always trust macros from this publisher.

Figure 4.1: An Excel spreadsheet contains macros signed by an untrusted publisher.

It's easy enough to just train users to click the "always trust" check box and then "Enable Macros," but that in itself creates an even larger and more troublesome problem: This dialog box is the *exact* feature intended to prevent malicious scripts and macros! By training users to essentially ignore it, you're bypassing important and beneficial safety measures within the product. A better solution is to *not* use self-signed certificates for code that you need to distribute to other people. Instead, obtain a *real* certificate from a CA that is mutually trusted by all involved parties.

You'll also see this problem on some Web sites or FTP sites where the administrators don't understand the difference between a "real" and "self-signed" certificate. Microsoft's Internet Information Services (IIS) 7.0 makes it easy for admins to create a self-signed SSL certificate, as Figure 4.2 shows. That's great for testing a site, but it won't create the mutual trust required on a public-facing site.


Internet Information Services (IIS) Manager
MYSERVER
File
View
Help
Connections
Start Page
MYSERVER (MYSERVER\Administrato
Application Pools
Sites
Default Web Site
Server Certificates
Actions
Import...
Create Certificate Request...
Complete Certificate Request...
Create Domain Certificate...
Create Self-Signed Certificate...
Help
Online Help
Features View
Content View
Ready

Use this feature to request and manage certificates that the Web server can use with Web sites configured for SSL.

Name	Issued To
	WMSvc-MYSERVER
ServerCert	ServerCert

Figure 4.2: Notice the option to "Create Self-Signed Certificate" in the right bar of this IIS management console.

Self-Signed CAs

A problem with similar roots but different manifestations are digital certificates that are issued by a self-signed root CA. Technically speaking, *all* root CAs have self-signed certificates—even commercial ones. That's why the root CA is where the issue of trust as we learned in previous chapters, begins—there's nobody else to vouch for the root CA. The problem arises when a company puts up their own Public Key Infrastructure (PKI) and self-signs their own root CA—rather than having a more globally-trusted CA sign their root, for example.

Okay, to be clear, tons of companies use self-signed CAs, and there's not really an inherent problem with that so long as the only people relying on the CA's certificates are inside the company or are partners of the company. If you work for Microsoft Corporation, for example, your company computer is probably pre-configured to trust certificates issued by Microsoft's internal PKI. No problem. The problem is when you ask someone *outside* of Microsoft to trust that CA. There are really two instances where this happens:

- A public Web or FTP server is assigned an SSL certificate that was signed by the company's internal, self-signed CA. This essentially requires everyone in the public to trust the company's internal CA—it's like having your Mom write you a letter that gives you permission to drive. It's nice of Mom, and we're all sure she did a great job of testing your driving skills and verifying your birthdate, but we'd much rather have someone independent (like the Department of Motor Vehicles) do that.
- Emails are signed or encrypted using certificates issued by an internal, self-signed CA. Again, that's great for strictly internal communications, but once the email leaves the company, you're asking the recipient to trust the company's CA. If the recipient works for a partner or a subsidiary, that might be reasonable; otherwise, the public would usually have a mutually-trusted third party validate your identity and issue you a certificate.

Really, this just brings the discussion back around to the all-important concept of *trust.* More importantly, *mutual trust.* When two parties are engaged in communications and want to be able to positively identify each other, they can't rely on self-made assertions of identity.

Imagine two strangers standing in a room trying to talk to one another but concerned about the actual identity of the other person. "I'm Bob," the first guy says, while the second says "I'm Jane." Both eye each other warily because they have no proof—just the other's say-so. But if they could both pull out photo identification from a mutually-trusted source—say, the Department of Motor Vehicles—then they could verify each other's identities and get on with the conversation.

A Web server that has an SSL certificate issued by the company's own self-signed CA is basically in the same situation. Imagine you visit www.theacmecompanies.com and plan to make a purchase. You see that the site is secured with an SSL certificate, and that the data channel is encrypted. But are you sure you're really talking to The ACME Companies, and not The Acme Companys or some other clever near-miss? The only way to check is to view the site's SSL certificate, which your browser lets you do. If it's issued from a self-signed, internal CA, that's basically saying, "I, The ACME Companies, swear that I am The Really Real ACME Companies. Honest." If, however, the certificate is issued by a CA that you happen to trust, then you're getting a third-party validation of the identity, and you can continue safely.

Let's Go Phishing

Phishing—the act of putting up a legitimate-looking Web site in order to gather sensitive information from a confused public—can really benefit from most users' misunderstanding of SSL certificates and trust.

Let's say a clever phisher puts up a Web site named www.amzon.com. It's an easy typo to miss if you were really headed to Amazon.com (which is why, in reality, Amazon themselves registered that mis-spelled domain name). If the phisher did a good job of duplicating the look of the legitimate Amazon.com Web site, he might fool someone into trying to make a purchase. With an SSL certificate, he could probably convince people to enter personal information such as credit card numbers.

Now, most reliable commercial CAs wouldn't issue an SSL certificate to this person because they'd be perfectly aware of what he was up to. But he'd be free to issue his own certificate from his own self-signed CA. Of course, users' browsers wouldn't trust that CA, so they'd display a warning message—which might alert users to the problem. And that's a big reason you shouldn't use certificates issued by your own, internal, self-signed CA in a public venue: Even if you're legitimate, you risk creating a situation that looks like an illegitimate one, and users will have very few means of telling the difference. Cautious ones might just walk away from your site.

Self-Signed: When It's Okay and When It's Not

Self-signed certificates and internal, self-signed root CAs have their purposes. A self-signed certificate is good for local testing, primarily development of software code. Internal, self-signed root CAs are great for strictly-internal uses, such as email and intranet applications, where all of the parties can be configured to trust the CA.

When you move beyond those uses, you move into a realm where *mutual* trust is important, and third-party verification of identity is an expected norm. In those instances, self-signed certificates are useless, and internal, self-signed root CAs can often be mistaken as illegitimate attempts to gain unwarranted trust.

Poor Identity Verification Processes

Honestly, the biggest difficulty for me in using certificates is this whole issue of trust. Pick a commercial CA, and tell me what process they use to verify identity before they issue a certificate. In many cases, it can be difficult to determine—and those are CAs that I often choose not to trust. I truly feel that a good commercial CA is not selling a packet of computer bits, they're selling *trust,* and a good one will advertise exactly how they create that trust by making it clear what their verification procedures are.

Take SSL or code-signing certificates, two certificate applications that have high stakes in terms of risk and trust. One commercial CA I'm familiar with offers "standard," "deluxe," and "premium" SSL certificates at varying price points. At the cheap end, they simply verify that you own the domain name—nothing more. They don't check to see that the domain name matches your company, that the company actually exists, or anything else. Think about that: If I have a Web site and can change its content, then I very clearly own or have control over the domain name. Why pay any amount of money just to have someone else attest to my control? It's ridiculous: It's like answering the front door of your own house and seeing a salesperson who wants to sell you a certificate saying you live there.

At the high end, that same commercial CA sells much more expensive certificates that do require that you verify your domain ownership, corporate entity, and other factors. That's great, and it's exactly the sort of validation I'd hope a certificate would entail—but the problem is this: When I'm viewing a Web site in my browser, can I easily determine whether the certificate is a "standard" one or a "premium" one? They're issued by the same commercial CA; are they issued from different roots, or the same one? Are they distinguishable to the end user? With questions like that I might well decide not to trust anything from that CA because I risk being confused between a useful high-end certificate and the essentially useless low-end one.

Note
This is not to say that all CAs offering multiple price points for certificates are going to include essentially useless ones in their lineup. There are legitimate reasons to have multiple pricing tiers, including "wildcard" certificates (which can secure all hosts within a given domain), and so forth.

Also, note that pricier Extended Validation (EV) certificates—offered by most commercial CAs—typically require more exhaustive identity checks, and are necessary in order to get the "green address bar" visual indicator supported by newer Web browsers (see Figure 4.3). In fact, these pricier EV certificates are becoming a de facto standard for many high-stakes sites, such as banks and large e-commerce sites that want to help their customers ensure that they're in the right place.

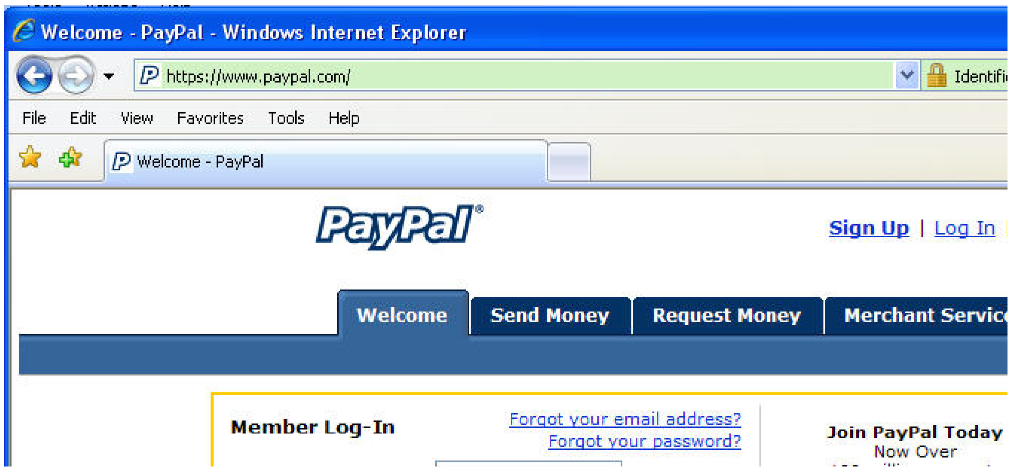


Figure 4.3: The "green address bar" indicates the use of an EV certificate.

EV Certificates

An EV certificate is just like any other digital certificate. However, the criteria for issuing these certificates was standardized by the Guidelines for Extended Validation Certificates (http://cabforum.org/EV_Certificate_Guidelines_V11.pdf), produced by the CA/Browser Forum, to help standardize identifying procedures across commercial CAs and provide a uniform measure of trust to consumers. Only certificates that are validated using these procedures can be marked as "EV," and Web browsers will only display the "green address bar" (or other symbology) for EV certificates. By allowing Web browsers to clearly differentiate between EV certificates and other certificates of unknown or lower quality, the Forum is helping more consumers become aware of the issue of trust, and helping to implement a standard for that trust across the industry.

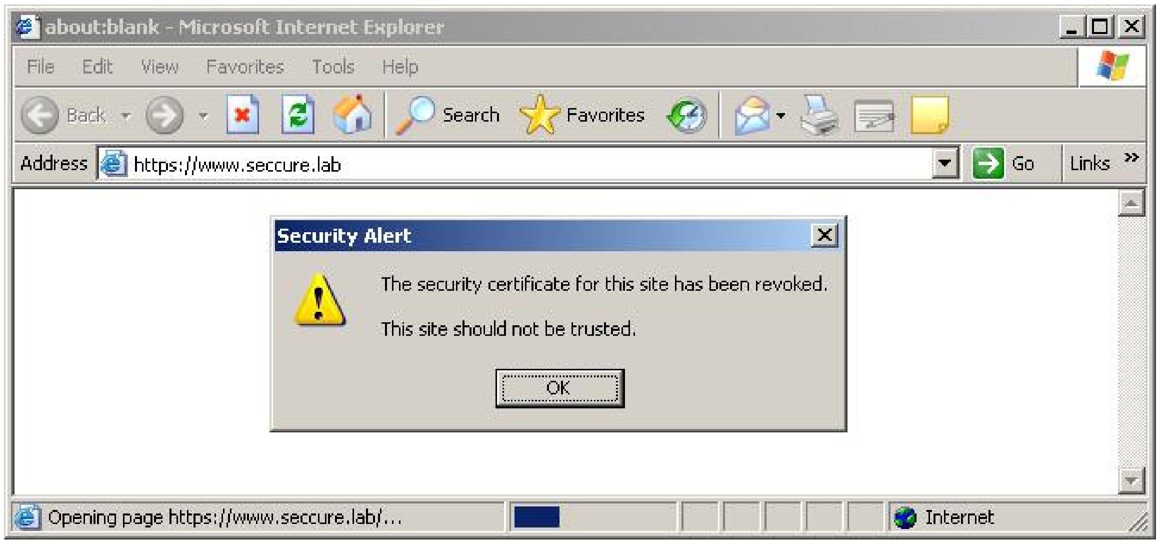
CAs offering EV certificates must pass an independent audit, and all EV-issuing CAs must follow the same minimum criteria. There's still room for individual CAs to go above and beyond, and most Web browsers will clearly display the name of the issuing CA in the "green address bar"—this helps consumers, for example, decide that they trust one CA company over another.

Confusion over identity vetting procedures, and the resulting differences in certificates from different CAs, definitely helps contribute to overall confusion about certificates. It's an opportunity for you to educate yourself and your users, and to decide what really earns your trust. EV certificates are fairly new, but they were designed in large part to help reduce the confusion between different CAs' policies and procedures by providing a minimum baseline for identity verification procedures.

Certificate Revocation: CRLs vs. OCSP

Here's where digital certificates really have an advantage over their physical cousins, such as driver's licenses: Revocation. Unlike a physical certificate, a digital one can be revoked by its issuing CA, rendering—in theory—that certificate useless. Of course, this depends on certificate clients—such as Web browsers and email programs—actually checking the revocation status of a certificate; not every client does that, simply because of the overhead required. You can't check a revocation unless you're connected to the network, for example, and doing so can take a couple of seconds.

Figure 4.4 shows what happens when a browser (Internet Explorer, in this case) encounters a revoked certificate. Here, the user has attempted to go to an https:// URL, which requires a certificate. The browser has checked the status of that certificate, realized it was revoked, and is warning the user.


about:blank - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
https://www.seccure.lab
Go
Links
Security Alert
OK
Opening page https://www.seccure.lab/...
Internet

The security certificate for this site has been revoked.

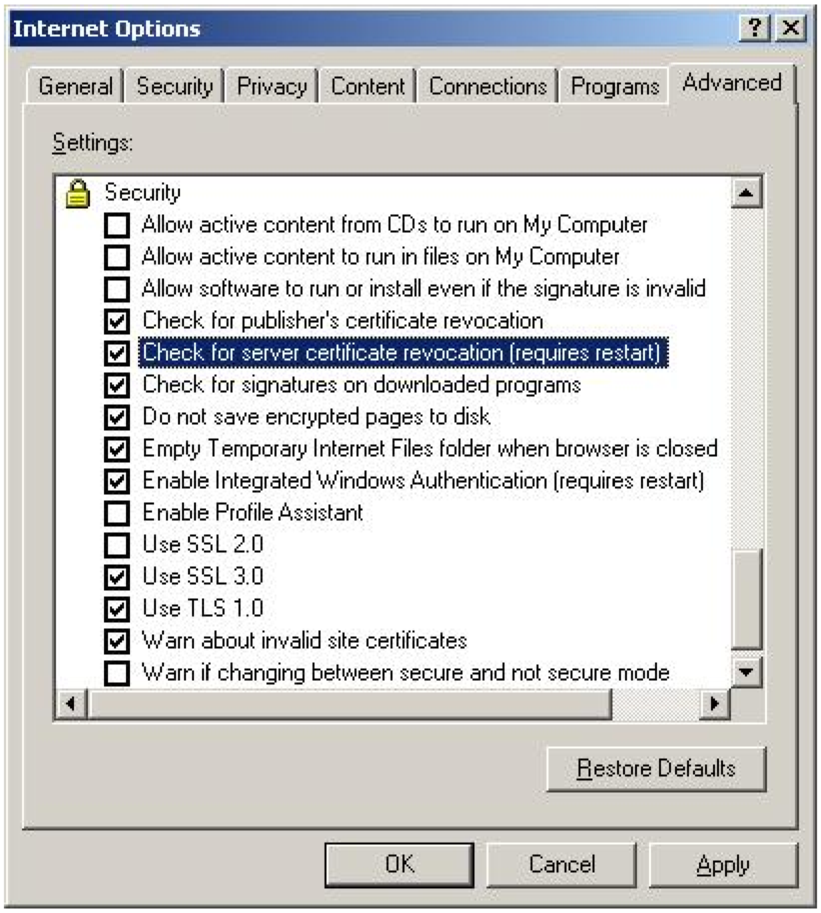
This site should not be trusted.

Figure 4.4: A revoked certificate generates a warning message.

Revoked or Expired?

Revocation is not precisely the same thing as *expiration;* all certificates eventually expire, and must be renewed. However, users tend to think of expiration as less “serious” that revocation: To them, revocation implies that someone (the issuing CA) had some reason to take action against you; expiration is something that happened passively because you forgot to renew. In reality, an expired certificate could also be one that *was revoked earlier,* has now passed its expiration date, but will not actually be renewed. Keep that in mind: A client won’t check the revocation status of an expired certificate, so you don’t ever know if the expired certificate is just lapsed and will be renewed soon (which does happen a lot), or if the expired certificate was previously revoked and shouldn’t be trusted. Expired certificates should be treated every bit as seriously as revoked ones—generally speaking, both expired and revoked certificates should be considered worthless and untrustworthy.

Most clients capable of checking certificates can also be configured to check for revocation. Figure 4.5 shows this configuration in a version of Internet Explorer.



Security
- [ ] Allow active content from CDs to run on My Computer
- [ ] Allow active content to run in files on My Computer
- [ ] Allow software to run or install even if the signature is invalid
- [x] Check for publisher's certificate revocation
- [x] Check for server certificate revocation (requires restart)
- [x] Check for signatures on downloaded programs
- [x] Do not save encrypted pages to disk
- [x] Empty Temporary Internet Files folder when browser is closed
- [x] Enable Integrated Windows Authentication (requires restart)
- [ ] Enable Profile Assistant
- [ ] Use SSL 2.0
- [x] Use SSL 3.0
- [x] Use TLS 1.0
- [x] Warn about invalid site certificates
- [ ] Warn if changing between secure and not secure mode

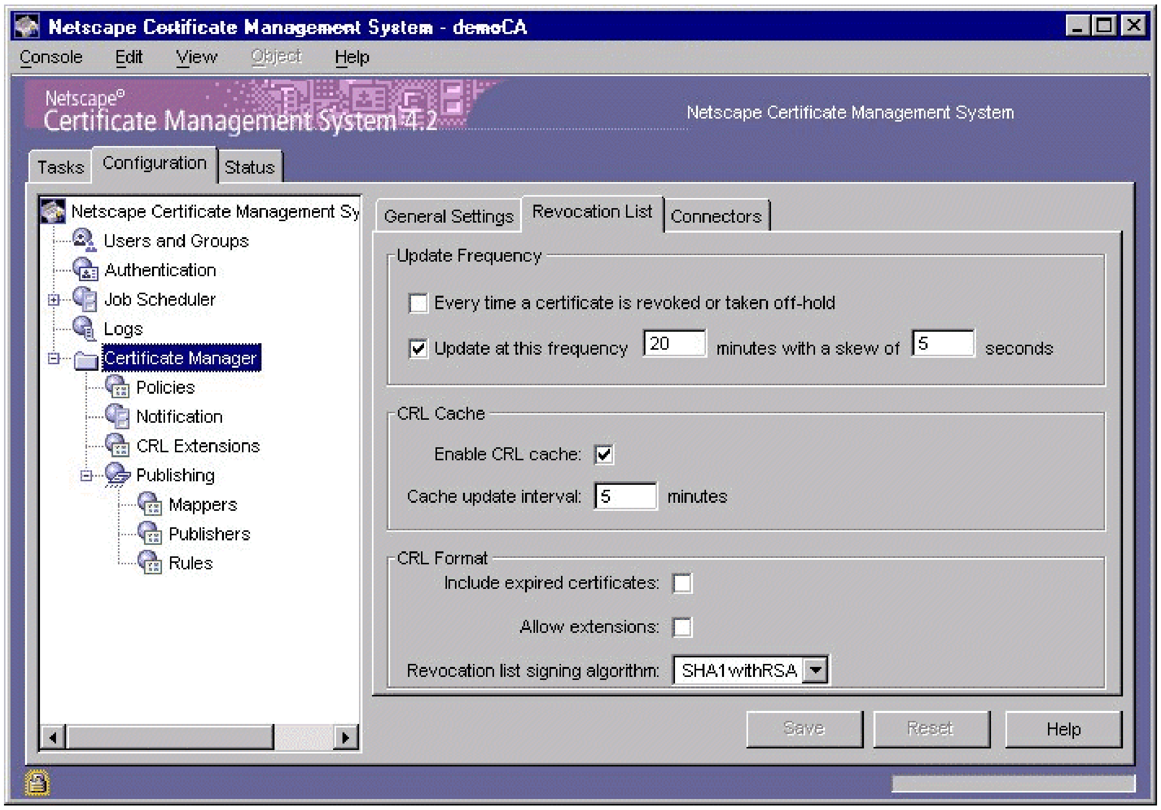
Figure 4.5: Configuring a check for certificate revocation.

So how do clients actually check for revocation? By one of two means: a CRL or OCSP.

CRL: Certificate Revocation List

In the good old days of certificates, a CRL was the accepted way to revoke a certificate. A CRL is basically a specially-formatted list, published by a CA, listing all the certificates that CA has revoked, along with the revocation reason. A revoked certificate should stay on the CRL until the certificate passes its natural expiration date, at which point the certificate is no longer valid anyway.

Reasons can include a compromise (loss or theft) of the certificate, or even a compromise of the CA itself, in which case all the CA's issued certificates may be revoked. Other reasons include a change in identity (such as a company name change), and so forth. CRLs are digitally signed using the CA's own certificate so that clients can easily verify the authenticity of the CRL. Almost any software capable of issuing certificates will also support CRL publication; Figure 4.6 shows an old Netscape Certificate Management system configuration dialog box, offering options for CRL publication. Note that some CRLs can include expired certificates, as shown in the options here.



Update Frequency

[ ] Every time a certificate is revoked or taken off-hold

[✓] Update at this frequency 20 minutes with a skew of 5 seconds

CRL Cache

Enable CRL cache: [✓]

Cache update interval: 5 minutes

CRL Format

Include expired certificates: [ ]

Allow extensions: [ ]

Revocation list signing algorithm: SHA1withRSA

Figure 4.6: Configuring a CRL in a CA.

CRLs have a very real-world counterpart: A couple of decades ago, telephone- or Internet-based credit card authorization still wasn't universal. Instead, credit card companies would publish quarterly or monthly lists of revoked credit card numbers, and prior to accepting a credit card, a merchant was supposed to see if the card was in that list. As you can imagine, looking up every credit card was a time-consuming process, and those printed bulletins became outdated pretty quickly. In a store with several cash registers, it was important to have several copies of the latest bulletins, or cashiers would have to wait on each other. Credit card companies would often publish interim bulletins every month, meaning you not only had to check the main bulletin but also any interim updates that had been published since—it was kind of a messy situation. And CRLs work in much the same way.

The server actually hosting a CRL has to be beefy because it may need to respond to millions of requests at any given moment. As CRLs grew, it became more and more difficult for servers to keep up with demand, and so some compromises and workarounds were developed: CRLs would be distributed across load-balanced servers, much like having multiple copies of a credit card bulletin in the store; servers would publish "delta CRLs," allowing clients to cache the main CRL locally, and then just consult the "most recent updates" CRL—not unlike the interim updates published by credit card companies.

Several years ago, various Internet working groups collaborated on a replacement for CRLs: OCSP.

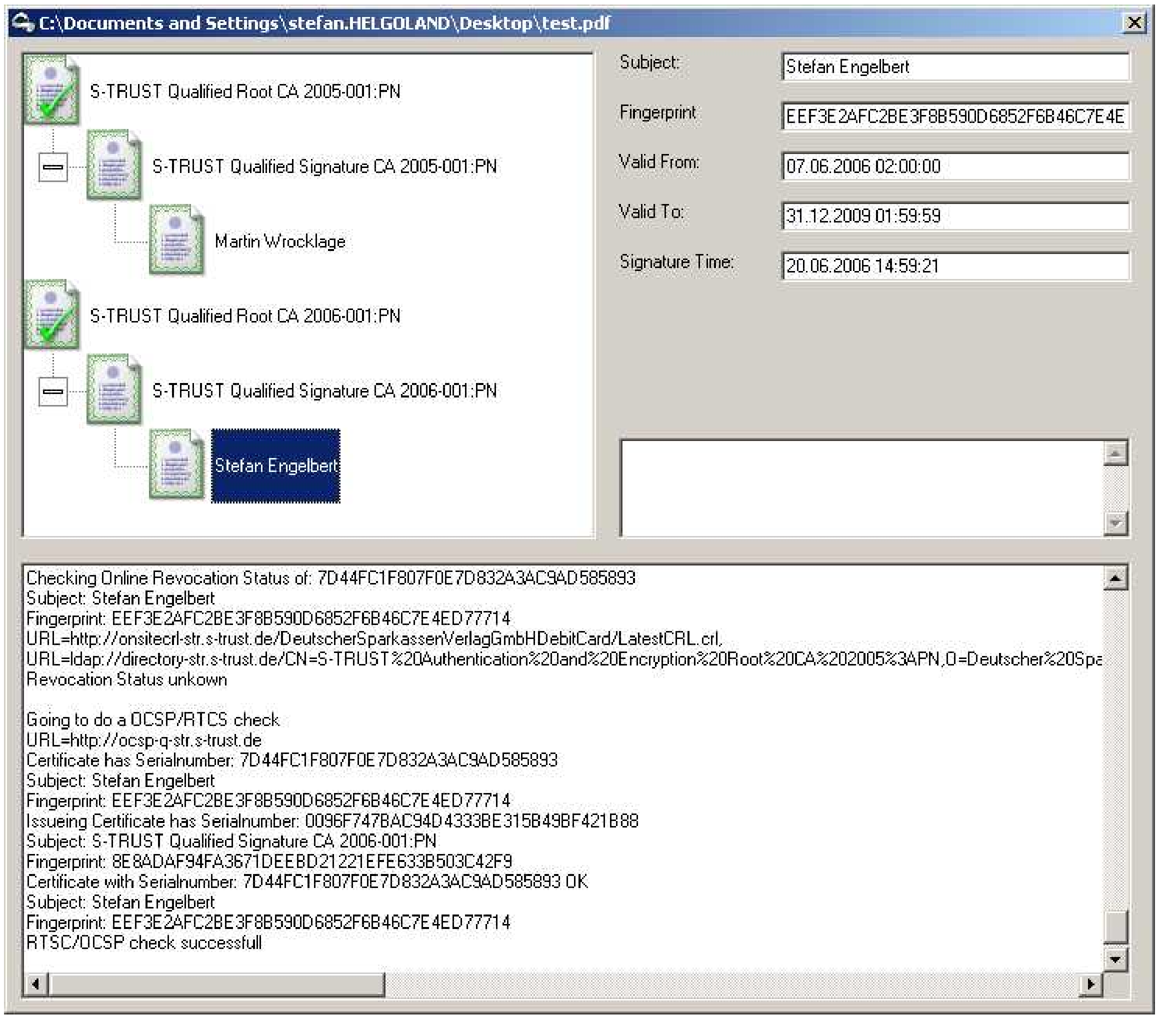
OCSP: Online Certificate Status Protocol

OCSP is designed to serve much the same purpose as a CRL, but it does so on a per-certificate basis. In other words, rather than downloading a whole list of revoked certificates, a client uses OCSP to simply check the status of the single certificate it's interested in at that moment. This is analogous to the now-widespread use of telephone- and Internet-based credit card authorization: Rather than reading through a bulletin of revoked card numbers, merchants today simply ask the credit card company in real-time whether the card they're looking at is valid.

OCSP solves most of the problems of CRLs: Because each OCSP response is small, fewer servers can handle larger workloads. Each OCSP response is for one certificate, so there's no need to publish delta lists and so forth, although clients are permitted to cache OCSP responses to save time in the near future.

OCSP gained widespread support in Internet Explorer 7 on Windows Vista. Firefox 3 enabled OCSP checking by default. Safari supports OCSP, as do later versions of Opera. Most newer Cisco devices and other network devices that rely on certificates also support OCSP.

OCSP isn't foolproof; using it requires clients to have a network connection to the CA's OCSP servers. In practice, this isn't usually difficult because the client is usually getting the certificate over a network connection right then, so making one more network connection for an OCSP check is possible. However, there are instances where certificates are stored locally—such as in the digital signatures on an application—where a client might want to verify them but might not have an available network connection at the time. Figure 4.7, for example, shows how Adobe's Acrobat Reader can use OCSP to check the certificate in a PDF document; in this example, a network connection is available to support the check—but what if you tried to do it on a laptop when a connection wasn't available? Still, OCSP is the best solution we have at the moment.



C:\Documents and Settings\stefan.HELGOLAND\Desktop\test.pdf

- S-TRUST Qualified Root CA 2005-001:PN
  - S-TRUST Qualified Signature CA 2005-001:PN
    - Martin Wrocklage
- S-TRUST Qualified Root CA 2006-001:PN
  - S-TRUST Qualified Signature CA 2006-001:PN
    - Stefan Engelbert

Subject:	Stefan Engelbert
Fingerprint	EEF3E2AFC2BE3F8B590D6852F6B46C7E4E
Valid From:	07.06.2006 02:00:00
Valid To:	31.12.2009 01:59:59
Signature Time:	20.06.2006 14:59:21

```
Checking Online Revocation Status of: 7D44FC1F807F0E7D832A3AC9AD585893
Subject: Stefan Engelbert
Fingerprint: EEF3E2AFC2BE3F8B590D6852F6B46C7E4ED77714
URL=http://onsitecrl-str.s-trust.de/DeutscherSparkassenVerlagGmbHDebitCard/LatestCRL.crl,
URL=ldap://directory-str.s-trust.de/CN=S-TRUST%20Authentication%20and%20Encryption%20Root%20CA%202005%3APN,O=Deutscher%20Spa
Revocation Status unkown

Going to do a OCSP/RTCS check
URL=http://ocsp-q-str.s-trust.de
Certificate has Serialnumber: 7D44FC1F807F0E7D832A3AC9AD585893
Subject: Stefan Engelbert
Fingerprint: EEF3E2AFC2BE3F8B590D6852F6B46C7E4ED77714
Issueing Certificate has Serialnumber: 0096F747BAC94D4333BE315B49BF421B88
Subject: S-TRUST Qualified Signature CA 2006-001:PN
Fingerprint: 8E8ADAF94FA3671DEEBD21221EFE633B503C42F9
Certificate with Serialnumber: 7D44FC1F807F0E7D832A3AC9AD585893 OK
Subject: Stefan Engelbert
Fingerprint: EEF3E2AFC2BE3F8B590D6852F6B46C7E4ED77714
RTSC/OCSP check successfull
```

Figure 4.7: Running an OCSP check.

The "Gotchas"

So where are the "gotchas" in CRLs and OCSP? First, simply being aware that certificates *can* be revoked, and that their status *should* be checked before trusting them, helps alleviate a major "gotcha" where a revoked certificate might be improperly trusted.

But understanding how CRLs can be worked around, and how OCSP can be defeated—by not having a network connection, for example—can also help alleviate "gotchas." You now know that some clients don't actually check for revocation by default, so you can make a decision to enable that, if it's appropriate for your situation.

Insource or Out? The Pros and Cons of Managing a PKI

Through most of this guide, we've discussed CAs in a fairly generic sense, and everything we've discussed applies to both commercial CAs as well as your own in-house PKI, if you have one. Many organizations look at the price of a digital certificate—close to $2000 from some CAs for an EV SSL certificate—and immediately assume that running their own, internal CA would be better—or at least cheaper. Sometimes it can be, sometimes not so much. There are really two major scenarios: Needing certificates strictly for internal use, and needing them for external use with the public.

Internal Use Only

If you're looking for a way to issue certificates strictly for internal use—such as signing internal applications, signing internal emails, encrypting internal files, and so forth, then an internal PKI is definitely a potential option. However, don't make the mistake of assuming that an internal PKI is entirely free of cost. Depending on the number of certificates you need, in fact, an internal PKI can wind up costing you more money in the long run than just buying commercially-issued certificates. Here are some considerations:

- You need to practice serious disaster recovery operations when you have your own PKI. Losing your root CA can render all your certificates invalid, so you need to make sure the root CA is protected.
- Serious security is also called for: If your root CA is compromised, your internal security is also compromised. Some organizations create their root CA, use it to authorize one or more subordinates, and then take the root CA completely off-line to better secure it.
- CA servers require maintenance, and in many cases, the server won't be suited to performing other tasks. It's not that running a CA is particularly intensive; it's just that it's a very security-sensitive operation, so it tends to be a standalone task. That means you'll be increasing the number of virtual or physical servers you have to keep patched and maintained.

Some best practices recommend that your PKI consist of, at minimum, two servers: a root and a subordinate. There's obviously maintenance costs associated with these, in addition to operating system (OS0 and software licensing costs, so that's the typical minimum cost of entry. If you're spending that just to issue yourself a couple of code-signing certificates, going with a commercial CA would probably be less expensive in the long run.

Larger organizations that issue thousands of certificates often find value in hosting their own PKI. They typically have dozens of CA servers, but the incremental cost of a dozen servers in an organization that already has thousands isn't that high, and it's offset by the cost savings from the thousands of certificates they issue and maintain.

External Use

It's hard to make an argument that an entirely-internal PKI is a good solution for producing certificates that must be trusted by the public. As we've already discussed in this chapter, attesting to your own identity isn't exactly going to generate a lot of trust with users and customers. Most companies who maintain an internal PKI for internal use also continue to purchase commercially-issued certificates for their public Web sites and any other certificate uses that involve public interaction.

Other PKI Considerations

Remember, a PKI is more than just a server. Here are some other considerations you'll have to keep in mind when deciding whether or not to host your own PKI:

- Revocation. You'll need to maintain enough CRL or OCSP servers—or potentially both, if you have clients that don't support OCSP—to handle revocation checks.
- Disaster Recovery. It's critical that every CA be backed up thoroughly, and that you practice disaster recovery procedures frequently to make sure your backups work.
- Process. You'll need to come up with a trustworthy process for identity verification—you can't just hand out certificates via email and expect them to remain secure.
- Trust. You'll need to configure your clients to trust your internal CA; in some cases, that can be accomplished centrally, but in others, it will require per-client manual configuration.
- Connectivity. If your clients won't all be connected to the same network as your CAs—mobile users, for example, won't always be—then you need to consider how they'll contact the CA for certificate retrieval, revocation checks, and so forth.

A PKI can be more complicated in reality than it might seem in theory, simply for security concerns. For example, Figure 4.8 shows a Microsoft-recommended way of setting up a PKI: An Issuing CA—that is, one which issues certificates—is located on the main corporate network. A standalone PKI network hosts the security-sensitive root CA and a Policy CA (which in Microsoft's world contains rules that determine who can have certificates, amongst other things). Thus, you're building out a whole new network segment, new servers, and so forth, including all the necessary storage and disaster recovery provisions (pictured as part of the Hierarchical Storage Management, or HSM, network in the diagram).

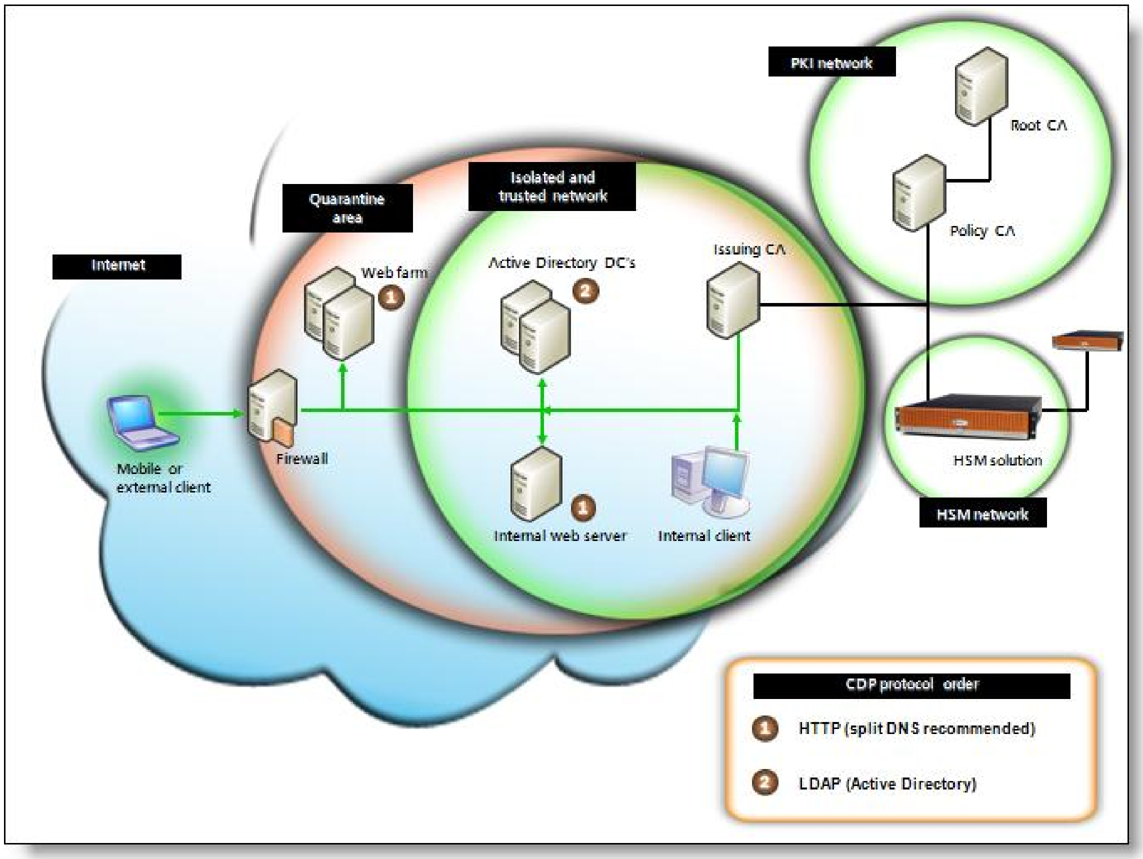


Figure 4.8: Microsoft-recommended design for an internal PKI.

Most major OSs now include or offer support for hosting a PKI: Windows Server, for example, has included a Certificate Services feature for close to a decade. The easy availability of the PKI software has led many organizations to implement fairly halfhearted PKI designs; in most cases, they did so just to issue "one or two" certificates, but then came to rely on that PKI more and more as they discovered what else it could do for them. In the end, most wind up going through painful and expensive re-architecture to produce a PKI actually capable of supporting their requirements.

This isn't meant to drive you away from considering your own internal PKI, but it is intended to help make sure that you have the right reasons for doing so, have reasonable expectations about what's really required, and are prepared for the expense and complexity that comes with an internal PKI. If you find that you're not—well, that's one reason commercial CAs exist.

Certificates in the Enterprise

We've covered a great deal of ground in these four chapters. We began with a look at what digital certificates are and how they work, including a look at asymmetric encryption and how it more or less enables all the great things that certificates can do. We've examined the certificate life cycle, from issuance through renewal or revocation. We've spent lots of time focused on *trust,* which is something that I think doesn't get enough attention in the world of certificates. Too few people really understand that the company *issuing* the certificate bears a tremendous responsibility for properly validating identity—in fact, too few people understand that certificates are *all about* identity, and that factors such as encryption really come as a side effect of that trust in identity. You, however, now understand all of that!

We've also looked at the many ways in which a simple certificate can be used to enhance security and trust throughout an organization—for e-commerce, email, data transfer, authentication and remote access, software security and the elimination of malware, and so forth. In many ways, certificates have always held the answers to some of our most challenging enterprise security needs—we just need to reach out and start utilizing certificates in every way possible.

I tried, in Chapter 3, to help you understand how important the concept of trust is to certificates as a whole. A certificate is literally a form of digital identification, and it's only as good as the organization that validated that identity and issued the certificate. We overlook that aspect too often, although new technologies such as EV certificates (and the corresponding Web browser support) are beginning to place more emphasis on the CA and on the concept of identity rather than just encryption.

Finally, in this chapter, we've explored some of the more hidden and tricky bits related to certificates, including why and when self-signed certificates can be useful, the ups and downs of managing your own internal PKI, how certificate revocation works, and—again—the importance of the identity verification process in maintaining the trustworthiness of a certificate. Hopefully, you've come to realize that not all certificates are created equal, and you'll be able to use that knowledge to make smart decisions about which certificates, from which CAs, you'll choose to trust in the future.

For years, I've truly felt that certificates were really an overlooked gold mine, relegated to providing e-commerce encryption and email signatures when they can, in fact, do so much more. Hopefully, you've got a better feel, now, for all the ways in which your own organization might be able to use certificates, and you can start solving security and identity challenges.

Further Reading

There's plenty more that can be said about digital certificates. Here's a short list of useful online resources that you might want to review to learn more:

- http://www.cren.net/crenca/onepagers/guidebook.html is an old but still relevant document that guides you through starting a digital certificates pilot project in your own organization.
- http://www.cren.net/crenca/pkiresources/index.html is an immense list of tech talks, papers, and other independent information and advice related to digital certificates and PKI.
- http://www.amazon.com/Understanding-PKI-Deployment-Considerations-Kaleidoscope/dp/0672323915 is an Amazon.com Web page for my favorite print book on PKI. If you decide to buy from them, be sure to check their certificate!
- Wikipedia has some well-written articles on various PKI topics; they're especially useful because they link to various reference materials such as official Internet drafts and other documents:
    - PKI: http://en.wikipedia.org/wiki/Public_key_infrastructure
    - CRL: http://en.wikipedia.org/wiki/Certificate_revocation_list
    - OCSP: http://en.wikipedia.org/wiki/Ocsp