

## Executive Summary

Several versions of Microsoft's IE browser do not check a certificate property known as a basic constraint to ensure that SSL server certificates have been issued by a certification authority (CA).

Anyone with a legitimate SSL server certificate and moderate technical knowledge may use that certificate to pose as a CA and issue SSL server certificates that will be accepted by these versions of IE.

## Discussion

The public key certificates defined in the original X.509 (1988) are all alike: there is nothing to distinguish a certificate belonging to a server or other user or end-entity from a certificate belonging to a certification authority (CA). A later version of X.509 added a certificate extension to serve this purpose.

Prudent applications check this extension to ensure that all server/user/end-entity certificates have been issued by a CA, and not by an end-entity posing as a CA. This extension is a basic constraint known as *isCA*.

Typically, when a CA issues its own certificate or issues certificates to other legitimate CAs it marks the *isCA* extension TRUE<sup>1</sup>; when it issues certificates to servers or users it marks the *isCA* extension FALSE (which is the default).

Before establishing a secure connecting to a server, prudent browsers check the value of *isCA* to make sure that the SSL certificate was issued by a CA (the CA's certificate has *isCA* set TRUE) and was issued to a server (the server's certificate has *isCA* set FALSE or absent; these are equivalent<sup>2</sup>):

TrustedRoot -> myServer

The browser trusts the TrustedRoot certificate because it is a built-in root (the CA in question reached an agreement with the browser manufacturer to have TrustedRoot included in the browser's database of trusted roots). The browser trusts myServer because a trusted root issued myServer's certificate.

Trust chains can be longer:

TrustedRoot -> aSubordinateCA -> myServer

Same principle: The browser trusts the TrustedRoot certificate because it is a built-in root. The browser trusts aSubordinateCA because a trusted root issued aSubordinateCA's CA certificate. And the browser trusts myServer because myServer's

---

<sup>1</sup> Typically, before a CA will issue a certificate to another CA, that is, before it will issue a certificate to a CA with *isCA* set to TRUE, it will ensure that the subordinate CA has identity validation procedures that meet some standard determined by the CA: TrustedRoot would not want aSubordinateCA to start issuing certificates to any old entity without some constraints on the certificate issued to aSubordinateCA (for example, TrustedRoot may include statements limiting liability in aSubordinateCA's certificate).

<sup>2</sup> There are also other constraints that indicate that a certificate belongs to a server, but a prudent reader may wish to question the treatment of constraints in IE, given the problems described in this paper.

certificate was issued by a trusted CA (aSubordinateCA, trusted because SubordinateCA's certificate was issued by a trusted root, see above)<sup>3</sup>.

A prudent browser will verify the value of *isCA* at all points, to ensure that an intermediate certificate belongs to a CA. In example above, the checks would be OK and the browser would trust myServer.

What if an attacker tried creating the chain

TrustedRoot -> myServer -> aBogusServer ?

For example, the owner of myServer could take the myServer certificate and associated private key, import them into an open source cryptography library, and issue a certificate that allows them to pose as [www.microsoft.com](http://www.microsoft.com):

TrustedRoot -> myServer -> [www.microsoft.com](http://www.microsoft.com)

A prudent browser will reject this, because before trusting [www.microsoft.com](http://www.microsoft.com) as a server, it must trust myServer as a CA, and when it checks the value of *isCA* in myServer's certificate, it will reject that certificate, since TrustedRoot set *isCA* to FALSE (or left it out, FALSE being the default) when issuing myServer's SSL certificate.

That is what prevents an attacker from creating the chain

TrustedRoot -> myServer -> [www.microsoft.com](http://www.microsoft.com)

and posing as [www.microsoft.com](http://www.microsoft.com) on the Internet. More to the point, and to be 100% factually correct, the attacker can issue all of the certificates they want, but prudent browsers will reject them, because my certificate has *isCA* set FALSE, thus preventing the attacker from posing as someone else.

Again, to be 100% correct, it is the browser that will reject the [www.microsoft.com](http://www.microsoft.com) certificate the attacker issued to themselves, since myServer is not a CA<sup>4</sup>.

**The problem: IE does not check *isCA*.** This allows anyone with a valid certificate issued by any TrustedRoot or by any trusted SubordinateCA to issue certificates to anyone they want and have those recognized as valid server certificates by IE. Remember, they don't have to a CA certificate, because IE is not checking the extension that determines whether a certificate is a CA certificate or not.

In practice, unless the attacker couples the bogus certificate with another attack, when you connect to a bogus server your browser would warn you that the DNS name of the server you are connecting to does not match the name in the certificate. For example, if

---

<sup>3</sup> There are many trusted roots whose certificates are built into your browser, and there are quite a few subordinate CAs, that is, legitimate CAs with legitimate CA certificates issued by trusted roots. For example, Verisign and Microsoft make extensive use of subordinate CAs to which their own trusted root has issued a CA certificate. Entrust.net is also such a CA; in fact, Entrust.net is both a trusted root in some browser versions and a subordinate CA, but that is a subject for another paper....

<sup>4</sup> Admittedly, as soon as an attacker acts as a CA using myServer's certificate, they likely violate the license agreement for that certificate they have with TrustedRoot, but if they are deliberately issuing bogus certificates, how likely am they to worry about a little old license agreement?

the attacker's server were aBogusServer.myServer.com<sup>5</sup>, its DNS name would not match the name in the certificate, which the attacker set as www.microsoft.com when they issued the certificate.

Unfortunately, IE does not prevent you from visiting this site, despite the mismatch. It asks you if you want to proceed. Many people will just click OK when the dialog because they want to get on with things. And users can disable that warning. Those are problems for another day, other papers....

Now, if the attacker could spoof the DNS or use other redirection techniques so that www.microsoft.com points to their server then there would be no mismatch and IE would allow the secure connection and you would think you were securely connected to Microsoft's web site. And DNS spoofing is possible (more difficult than issuing bogus certificates, but possible). And redirection tools are available on web sites catering to the hacker community.

### **Conclusion**

This flaw in IE is serious, as it permits attackers to masquerade their web sites as the sites of their legitimate owners. Some reviewers of this problem speculate that this attack may already have taken place in the field, since this flaw has existed in IE for quite some time.

### **Background**

More on the flaw, including a case study showing how easy it is to exploit, is available at: <http://theregister.co.uk/content/4/26620.html>

MS is downplaying this, but the open source community has already fixed an open source browser with the same problem: <http://www.theregister.co.uk/content/55/26653.html>

---

<sup>5</sup> This is the name in the URL that you clicked on, thinking you were connecting to Microsoft: the attacker labeled it as Microsoft, but underneath the covers it points to aBogusServer. (Unless the attacker can spoof the DNS, see below.)