



# Web Security for Business: Creating and Implementing a Private Certificate Authority with OpenSSL and mod\_ssl

Presented at O'Reilly Open Source  
Convention - 7.27.2001 - San Diego,  
California - by Paul Weinstein -  
Internet Systems Virtuoso - Red Hat,  
Inc. - <pdw@redhat.com>





# What You Should Know

How SSL/TLS works

Maintain and Run Apache, Apache  
Modules

CGI Interface works, know Perl

How to get around in Un\*x shell



# What We're Going to Talk About

## The Basics:

- How to create a private certificate authority (CA).
- How to sign server certificate request with private CA.
- How to sign and distribute client certificate request with private CA.

## The Nit and Gritty

- OpenSSL Configuration File
- Some HTML and Perl Code
- How to publish private CA within a limited environment.
- Configuring mod\_ssl to authenticate access based on client certificates issued by private CA.
- Certificate Revocation and Revocation Lists



# Disclaimer

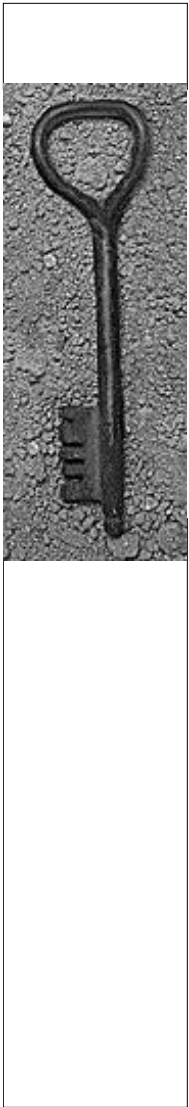
This presentation does not cover all of the security issues involved in maintaining a certificate authority (CA) or the data that is being protected by the CA.

Nor does this presentation cover all of the issues involved in securing a networked based machine and its contents, but only covers issues involved in securing and authenticating data transmitted between machines.

# Quick Review

Digital Certificates

Certificate Authorities





# Digital Certificates

## Type Of Digital Certificates

- Server Certificate
- Client Certificate

## X.509 Format, Issued By Certificate Authorities

- A Serial Number
- Name Of Issuing Certificate Authority
- Identifying Information, Such As; Name, Street Address and/or Email Address
- Subject's Public Key
- A “Signature” Of Issuing Certificate Authority



# Certificate Authorities

Public Certificate Authority; Verisign, Thawte, Equifax; recognized by default by most web browsers and web servers; used when no other relation exists between two parties.

Private Certificate Authority; by default not recognized; used when a relationship already exists between two parties.



# The Basics: OpenSSL

Creating our private certificate authority

Creating a Server Certificate Request

Signing a Server Certificate Request

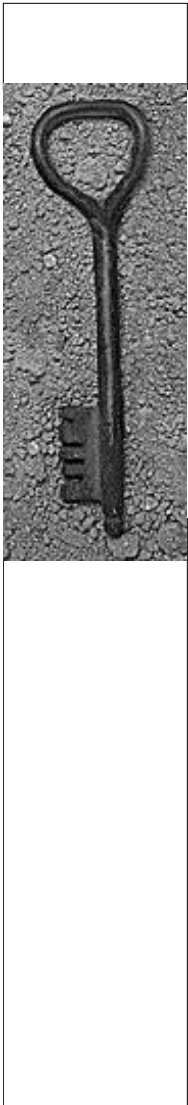
Signing a Client Certificate Request



# Creating a Private Certificate Authority

Creating a self-sign (root) certificate for private CA

```
bash-2.04$ cd /usr/local/openssl-0.9.6a/bin
bash-2.04$ ./openssl req -new -x509 -keyout ../ssl/private/ca.key -out ../ssl/certs/ca.cert -config ../ssl/openssl.cnf
Using configuration from ../ssl/openssl.cnf
Generating a 1024 bit RSA private key
.....++++++
.....++++++
writing new private key to '../ssl/private/ca.key'
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:California
Locality Name (eg, city) []:Oakland
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Red Hat, Inc.
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:Paul Weinstein
Email Address []:pdw@redhat.com
bash-2.04$ █
```



# Creating a Certificate Signing Request (CSR) for Server

Since Apache uses OpenSSL via `mod_ssl` for SSL we'll use it to create a CSR for Apache.

```
bash-2.04$ ./openssl req -new -keyout ../ssl/private/server.key -out ../ssl/certs/server.csr -days 365 -config ../ssl/openssl.cnf
Using configuration from ../ssl/openssl.cnf
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to '../ssl/private/server.key'
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:California
Locality Name (eg, city) []:Oakland
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Red Hat, Inc.
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:Paul Weinstein
Email Address []:pdw@redhat.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
bash-2.04$ █
```



# Signing Our Server CSR

Now we'll sign this CSR using OpenSSL and our Private Certificate Authority.

```
bash-2.04$ ./openssl ca -out ../ssl/certs/server.cert -config ../ssl/openssl.cnf
-infiles ../ssl/certs/server.csr.1
Using configuration from ../ssl/openssl.cnf
Enter PEM pass phrase:
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName      :PRINTABLE:'US'
stateOrProvinceName  :PRINTABLE:'California'
localityName     :PRINTABLE:'Oakland'
organizationName  :PRINTABLE:'Red Hat, Inc.'
commonName      :PRINTABLE:'Paul Weinstein'
emailAddress     :IA5STRING:'pdu@redhat.com'
Certificate is to be certified until May 31 07:37:29 2002 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
bash-2.04$ █
```



# The Wonderful World Of Web Browsers

Different Web Browsers Support Different Methods For Creating Client Certificates.

The General Procedure:

- User Access a Web Page With Their Favorite Client (Web Browser).
- User Enters Identification Information Into Web Page Form.
- Submit Form Which:
  - Has Client Generate A Public And Private Key.
  - CGI Script Adds Public Key To Identification Information Being Submitted, Which Creates A Client Certificate Signing Request



# Signing Our Client CSR

After A Little Magic We Can Sign This Client CSR.

```
bash-2.04$ ./openssl ca -spkac ../ssl/certs/client.csr -out ../ssl/certs/client.
cert -days 365 -config ../ssl/openssl.cnf
Using configuration from ../ssl/openssl.cnf
Enter PEM pass phrase:
Check that the SPKAC request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
commonName      :PRINTABLE:'Employee Certificate for Paul Weinstein'
emailAddress    :IASSTRING:'pdw@redhat.com'
organizationName :PRINTABLE:'Red Hat, Inc.'
organizationalUnitName:PRINTABLE:''
localityName    :PRINTABLE:'Oakland'
stateOrProvinceName :PRINTABLE:'California'
countryName     :PRINTABLE:'US'
Certificate is to be certified until May 31 20:10:29 2002 GMT (365 days)

Write out database with 1 new entries
Data Base Updated
bash-2.04$
```



# The Nit And Gritty

What's In openssl.cnf File

HTML Code For Submit Form

Perl Scripts For Creating Client Certificate  
Signing Request and Installing Client Certificate  
Into Browser

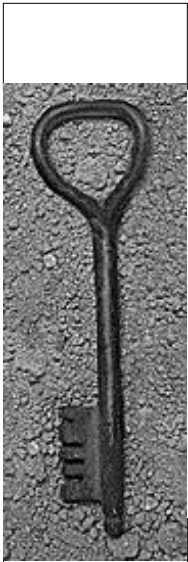
Publishing private CA within a limited  
environment.

Configuring mod\_ssl to authenticate access based  
on client certificates issued by private CA.

Certificate Revocation and Revocation Lists

# openssl.cnf

## Our Self-Signed Certificate and Private Key

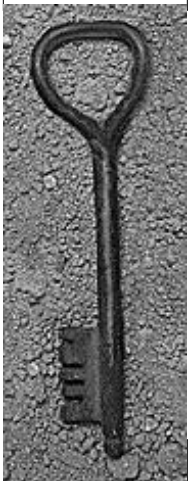


```
[ CA_default ]
dir               = /usr/local/openssl-0.9.6a/ssl           # Where
everything is kept
certs             = $dir/certs                            # Where the issued certs are kept
crl_dir          = $dir/crl                              # Where the issued crl are kept
database         = $dir/index.txt                        # database index file.
new_certs_dir    = $dir/certs                            # default place for new certs.

certificate      = $dir/certs/ca.cert                    # The CA certificate
serial          = $dir/serial                            # The current serial number
crl              = $dir/crl.pem                          # The current CRL
private_key      = $dir/private/ca.key                   # The private key
RANDFILE        = $dir/private/.rand                    # private random number file

x509_extensions = usr_cert                              # The extensions to add to the cert
```

# openssl.cnf



```
#####
[ req ]
default_bits          = 1024
default_keyfile       = privkey.pem
distinguished_name    = req_distinguished_name
attributes            = req_attributes
x509_extensions      = v3_ca # The extensions to add to the self signed cert
...

[ req_distinguished_name ]
countryName           = Country Name (2 letter code)
countryName_default   = AU
countryName_min       = 2
countryName_max       = 2

stateOrProvinceName   = State or Province Name (full name)
stateOrProvinceName_default = Some-State

localityName           = Locality Name (eg, city)

0.organizationName    = Organization Name (eg, company)
0.organizationName_default = Internet Widgets Pty Ltd

# we can do this but it is not needed normally :-
#1.organizationName    = Second Organization Name (eg, company)
#1.organizationName_default = World Wide Web Pty Ltd

organizationalUnitName = Organizational Unit Name (eg, section)
#organizationalUnitName_default =

commonName             = Common Name (eg, YOUR name)
commonName_max         = 64

emailAddress           = Email Address
emailAddress_max       = 40
```



# openssl.cnf

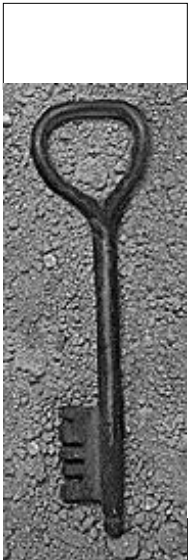
Defining a policy of necessary information in our certificates

```
policy                = policy_match

# For the CA policy
[ policy_match ]
countryName           = match
stateOrProvinceName  = match
organizationName      = match
organizationalUnitName = optional
commonName            = supplied
emailAddress          = optional

# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.
[ policy_anything ]
countryName           = optional
stateOrProvinceName  = optional
localityName         = optional
organizationName      = optional
organizationalUnitName = optional
commonName            = supplied
emailAddress          = optional
```

# HTML Form

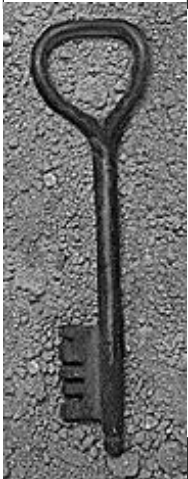


```
<HTML><HEAD><TITLE>Create Client Certificate</TITLE></HEAD><BODY>
<CENTER><H1>Create Client Certificate</H1></CENTER>

<FORM NAME="GenerateForm" ACTION="process-netscape.cgi">
<TABLE>
<TR><TD>Common Name:</TD><TD>
<INPUT TYPE="TEXT" NAME="commonName" VALUE="Employee Certificate for" SIZE=64>
</TD></TR>
<TR><TD>email:</TD><TD>
<INPUT TYPE="TEXT" NAME="emailAddress" VALUE="" SIZE=40>
</TD></TR>
<TR><TD>Organization:</TD><TD>
<INPUT TYPE="TEXT" NAME="organizationName" VALUE="Red Hat, Inc.">
</TD></TR>
<TR><TD>Organizational Unit:</TD><TD>
<INPUT TYPE="TEXT" NAME="organizationalUnitName" VALUE="Secure Services">
</TD></TR>
<TR><TD>Locality (City):</TD><TD>
<INPUT TYPE="TEXT" NAME="localityName" VALUE="Oakland">
</TD></TR>
<TR><TD>State:</TD><TD>
<INPUT TYPE="TEXT" NAME="stateOrProvinceName" VALUE="California">
</TD></TR>
<TR><TD>Country:</TD><TD>
<INPUT TYPE="TEXT" NAME="countryName" VALUE="US" SIZE="2">
</TD></TR>
</TABLE>
<!--
keygen is Mozilla specific and will be ignored in
internet explorer
-->
<KEYGEN NAME="SPKAC" CHALLENGE="challengePassword">

<INPUT TYPE="SUBMIT" NAME="SUBMIT">
</FORM>
<P><HR></BODY></HTML>
```

# HTML Form



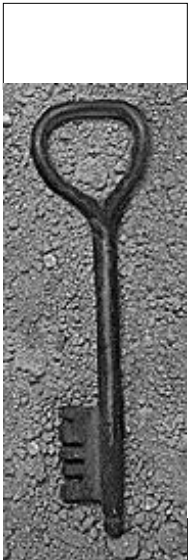
The screenshot shows a Mozilla browser window with the title "Create Client Certificate - Mozilla (Build ID: 2001050518)". The main content area displays a form titled "Create Client Certificate" with the following fields:

- Common Name:
- email:
- Organization:
- Organizational Unit:
- Locality (City):
- State:
- Country:

Below the fields is a "Submit Query" button. At the bottom of the browser window, the status bar shows "Document Done (0.561 sec)" and the Mozilla logo.

# CGI & Perl

## Setup our environment



```
#!/usr/bin/perl -U

use strict;
use CGI;

use File::CounterFile;
# module to maintain certificate request counter

my $base_dir = "/usr/local/openssl-0.9.6a/ssl";

my $query = new CGI;
# get a handle on the form data

my $key = $query->param('SPKAC');
# this will fail if not Mozilla-based browser
unless($key) { fail("No Key provided $key. Mozilla-based browser required"); }

my $counter = new File::CounterFile("$base_dir/serial", 1);
unless($counter) { fail("Could not create counter: $!"); }
my $count = $counter->inc();

my $req_file = "$base_dir/cert$count.req"; # certificate request filename
my $result_file = "$base_dir/cert$count.result"; # certificate filename

# Explicitly list form fields we must have for certificate creation to work.
my @req_names = ('commonName', 'emailAddress', 'organizationName',
                 'organizationalUnitName', 'localityName', 'stateOrProvinceName',
                 'countryName', 'SPKAC');
```



# CGI & Perl

Save the submit data and create a client CSR file.

```
# build the request file
open(REQ, ">$req_file") or fail("Could not create request $req_file: $!");

my $name;
foreach $name (@req_names) {
    my $value = $query->param("$name");
    $value =~ tr/\n//d;
    print REQ "$name = $value\n";
}
close(REQ);
print $query->header;
print $query->start_html(-title=>"Client Certificate Request",
                        -bgcolor=>"white");
print <<EOF;

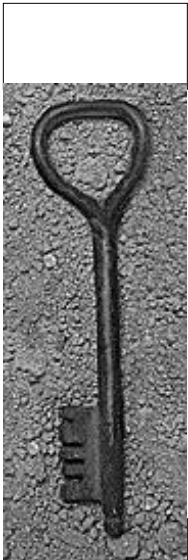
<FONT SIZE=+2 FONT=Helvetica>
Your certificate request has been sent. Your new cert will be
sent soon.

EOF

print $query->end_html;
exit(0);
```

# CGI & Perl

## Getting a signed certificate to client



```
#!/usr/bin/perl
use CGI;
$query = new CGI;
$cert = $ARGV[0];
$certdir = "/usr/local/openssl-0.9.6a/ssl/certs";
$result_file = "$certdir/$cert.result";
open(CERT, "<$result_file") || die "Can't open file $result_file";
# send the client certificate to the browser
print "Content-Type: application/x-x509-user-cert\n";
$result = join '', <CERT>;
close CERT;
$len = length($result);
print "Content-Length: $len\n\n";
print $result;
exit(0);
```



# Apache and mod\_ssl

Setup Apache so clients can download CA's root certificate.

```
#  
# Some MIME-types for downloading Certificates and CRLs  
#  
AddType application/x-x509-ca-cert .cer  
AddType application/x-pkcs7-crl .crl
```



# Apache and mod\_ssl

## Configuring our server certificate for Apache

```
# Server Certificate:
# Point SSLCertificateFile at a PEM encoded certificate. If
# the certificate is encrypted, then you will be prompted for a
# pass phrase. Note that a kill -HUP will prompt again. A test
# certificate can be generated with `make certificate' under
# built time. Keep in mind that if you've both a RSA and a DSA
# certificate you can configure both in parallel (to also allow
# the use of DSA ciphers, etc.)
SSLCertificateFile /usr/local/openssl-0.9.6a/ssl/certs/server.cert

# Server Private Key:
# If the key is not combined with the certificate, use this
# directive to point at the key file. Keep in mind that if
# you've both a RSA and a DSA private key you can configure
# both in parallel (to also allow the use of DSA ciphers, etc.)
SSLCertificateKeyFile /usr/local/openssl-0.9.6a/ssl/private/server.key
```



# Apache and mod\_ssl

Adding our CA to Apache and using it to authenticate clients

```
# Certificate Authority (CA):
# Set the CA certificate verification path where to find CA
# certificates for client authentication or alternatively one
# huge file containing all of them (file must be PEM encoded)
# Note: Inside SSLCertificatePath you need hash symlinks
#       to point to the certificate files. Use the provided
#       Makefile to update the hash symlinks after changes.
SSLCertificateFile /usr/local/openssl-0.9.6a/ssl/certs/ca.cert

# Client Authentication (Type):
# Client certificate verification type and depth. Types are
# none, optional, require and optional_no_ca. Depth is a
# number which specifies how deeply to verify the certificate
# issuer chain before deciding the certificate is not valid.
SSLVerifyClient require
#SSLVerifyDepth 10
```



# Certificate Revocation

Revoking a certificate before it expires and creating a certificate revocation list.

```
bash-2.04$ ./openssl ca -revoke ../ssl/certs/03.pem -config ../ssl/openssl.cnf
Using configuration from ../ssl/openssl.cnf
Enter PEM pass phrase:
Revoking Certificate 03.
Data Base Updated
bash-2.04$ ./openssl ca -genctrl -out ../ssl/crl/crl.pem
Using configuration from /usr/local/openssl-0.9.6a/ssl/openssl.cnf
Enter PEM pass phrase:
bash-2.04$
```



# Certificate Revocation

Making sure our Apache server doesn't accept the revoked certificate.

```
# Certificate Revocation Lists (CRL):  
# Set the CA revocation path where to find CA CRLs for client  
# authentication or alternatively one huge file containing all  
# of them (file must be PEM encoded)  
# Note: Inside SSLCARevocationPath you need hash symlinks  
#       to point to the certificate files. Use the provided  
#       Makefile to update the hash symlinks after changes.  
#SSLCARevocationPath /usr/local/sh3-3012/conf/ssl.crl  
SSLCARevocationFile /usr/local/openssl-0.9.6a/ssl/crl/crl.pem
```



## Citation

Engelschall, Ralf *User Manual mod\_ssl Version 2.8*  
30 Jan. 2001 <<http://www.modssl.org/docs/2.8/>>

Hirsch, Frederick *Introducing SSL and Certificates using SSLeay* Summer 1997  
<<http://www.ultranet.com/~fhirsch/Papers/wwwj/article.html>>

Leach, Mike and Tim Starr *Using Certificate Revocation Lists* 12 Dec. 2000  
<<http://www.apacheweek.com/issues/00-12-22>>



# Acknowledgements & Suggested References

Red Hat's Stronghold Team

This presentation:

- [http://www.weinstein.org/redhat/presentations/oscon01/intro\\_to\\_private\\_ca](http://www.weinstein.org/redhat/presentations/oscon01/intro_to_private_ca) (HTML)
- [http://www.weinstein.org/redhat/presentations/oscon01/intro\\_to\\_private\\_ca.pdf](http://www.weinstein.org/redhat/presentations/oscon01/intro_to_private_ca.pdf) (PDF)

Apache Project, <http://www.apache.org>

Apache Week, <http://www.apacheweek.com>



# Acknowledgements & Suggested References

mod\_ssl Project, <<http://www.modssl.org>>

- Mailing Lists, List Archives:
  - <[modssl-announce@modssl.org](mailto:modssl-announce@modssl.org)>
  - <[modssl-users@modssl.org](mailto:modssl-users@modssl.org)>
  - <<http://marc.theaimsgroup.com/?l=apache-modssl>>

OpenSSL Project, <<http://www.openssl.org>>

- Mailing Lists, List Archives:
  - <[openssl-announce@openssl.org](mailto:openssl-announce@openssl.org)>
    - <<http://marc.theaimsgroup.com/?l=apache-modssl> >
  - <[openssl-cvs@openssl.org](mailto:openssl-cvs@openssl.org)>
    - <<http://www.progressive-comp.com/Lists/?l=openssl-cvs>>
  - <[openssl-dev@openssl.org](mailto:openssl-dev@openssl.org)>
    - <<http://www.progressive-comp.com/Lists/?l=openssl-dev>>
  - <[openssl-users@openssl.org](mailto:openssl-users@openssl.org)>
    - <<http://www.progressive-comp.com/Lists/?l=openssl-users>>