New Tricks For Defeating SSL In Practice

Moxie Marlinspike
moxie@thoughtcrime.org
The Back Story
SSL And Certificate Chaining
This certificate has been verified for the following uses:

SSL Server Certificate

Issued To

Common Name (CN)      www.paypal.com
Organization (O)      PayPal, Inc.
Organizational Unit (OU) Information Systems

Issued By

Common Name (CN)      VeriSign Class 3 Extended Validation SSL SGC CA
Organization (O)      VeriSign, Inc.
Organizational Unit (OU) VeriSign Trust Network

Validity

Issued On            05/01/2008
Expires On            05/02/2009

Fingerprints

You probably know what they do...
More specifically...
CA Certificate
Embedded in browser.
All powerful.
Certifies that a site certificate is authentic.

Site Certificate
Identifies a particular URL
Is known to be authentic based on CA Certificate's signature.
CA Certificate
Embedded in browser.
All powerful.
Certifies that an intermediate CA is authentic.

Intermediate CA
Not embedded in browser.
Still sort of all-powerful.
Certifies that a site certificate is authentic.

Site Certificate
Identifies a particular URL.
Is known to be authentic based on CA Certificate's signature.
Certificate Chains Can Be > 3

Root CA → Intermediate → Intermediate → Intermediate → Leaf
How do we validate these things?
Almost everyone tells you the same story.
What they say:

Verify that the leaf node has the name of the site you're connecting to.
Verify that the leaf node hasn't expired.
Check the signature.
If the signing certificate is in our list of root CA's, stop.
Otherwise, move one up the chain and repeat.
Here Be Dragons

Very tempting to use a simple recursive function.

Everyone focuses on the signature validation.

The result of a naïve attempt at validation is a chain that is complete, but nothing more.
What if...

Root CA

Intermediate

Intermediate

Leaf (blueanarchy.org)
What if...

Root CA

Intermediate

Intermediate

Leaf (blueanarchy.org)

Leaf (paypal.com)
What they say:

Verify that the leaf node has the name of the site you're connecting to.
Verify that the leaf node hasn't expired.
Check the signature.
If the signing certificate is in our list of root CA's, stop.
Otherwise, move one up the chain and repeat.
Something must be wrong, but...

All the signatures are valid.
Nothing has expired.
The chain is in tact.
The root CA is embedded in the browser and trusted.
But we just created a valid certificate for PayPal, and we're not PayPal?
The missing piece...
...is a somewhat obscure field.
Back In The Day

Most CA's didn't explicitly set basicConstraints: CA=FALSE

A lot of web browsers and other SSL implementations didn't bother to check it, whether the field was there or not.

Anyone with a valid leaf node certificate could create and sign a leaf node certificate for any other domain.

When presented with the complete chain, IE, Konqueror, OpenSSL, and others considered it valid.
And then in 2002...

Microsoft did something particularly annoying, and I blew this up by publishing it.

Microsoft claimed that it was impossible to exploit.

So I also published a tool that exploits it.
sslsniff
sslsniff

Client Side:
Intercepts HTTPS traffic.
Generates a certificate for the site the client is connecting to.
Signs that with whatever certificate you specify.
Proxies data through.

Server Side:
Makes normal HTTPS connection to the server.
Sends and receives data as if it's a normal client.
Back before people started checking BasicConstraints:
All you had to do was pass sslniff a valid leaf node certificate for any domain.
It would automatically generate a certificate for the domain the client was connecting to on the fly.
It would sign that certificate with the leaf node.
IE, Konqueror, etc... wouldn't notice the difference.
You'd be surprised who still doesn't check basic constraints.

Even when people got warning dialogs in browsers that had been fixed, most of the time they'd just click through them.

Still useful as a general MITM tool for SSL.

The folks who did the MD5 hash collision stuff used sslsniff to hijack connections once they'd gotten a CA cert.

There are other uses yet, to be disclosed another day.
Surely we can do better.
The things you learn in TV studios.
The things you learn in TV studios.
The things you learn in TV studios.
The things you learn in TV studios.

This button posts to an HTTPS link, but there's no way to know that.

It's a button, so if you mouse-over it, the link isn't displayed in the browser bar at the bottom. The best you could do would be to view the page source, but that's problematic in browsers like Firefox that issue a second request to the server for the source.
Still prevalent today...
Still prevalent today...
There are some generalizable attacks here.
Browsers Then And Now...
Then: A Positive Feedback System

A number of indicators deployed to designate that a page *is* secure.

A proliferation of little lock icons.

URL bars that turn gold.
Then: An example from Firefox 2
Then: An example from Firefox 2

https://login.yahoo.com/config/mail?intl=us
Then: An example from Firefox 2
Then: An example from Firefox 2
Now: A Negative Feedback System

Less emphasis on sites being secure.
The proliferation of little locks has been toned down.
Firefox's gold bar is gone.
More emphasis on alerting users to problems.
A maze of hoops that users have to jump through in order to access sites with certificates that aren't signed by a CA.
Now: An example from Firefox 3

Yahoo! Mail: The best web-based email! - Mozilla Firefox

Chat right from your Inbox.
» See how

Your inbox understands you’ve got news to share.

See which of your contacts are online at a glance.

Easily switch from email to chat and back again.

Start right away, no download or setup needed.

See how to instantly reach friends and family from the New Yahoo! Mail.
Now: An example from Firefox 3

https://login.yahoo.com/config/login_verify2?&.src=ym

Yahoo! Mail: The best web-based email! - Mozilla Firefox

Chat right from your Inbox.
» See how

Your inbox understands you've got news to share.

See which of your contacts are online at a glance.

Easily switch from email to chat and back again.

Start right away, no download or setup needed.

See how to instantly reach friends and family from the New Yahoo! Mail.
Now: An example from Firefox 3

https://login.yahoo.com/config/login_verify2?&.src=ym

Chat right from your Inbox.
» See how

Your inbox understands you’ve got news to share.

See which of your contacts are online at a glance.

Easily switch from email to chat and back again.

Start right away, no download or setup needed.

See how to instantly reach friends and family from the New Yahoo! Mail.
Now: An example from Firefox 3
Secure Connection Failed

www.riseup.net uses an invalid security certificate.

The certificate is not trusted because the issuer certificate is unknown.
The certificate is only valid for admin.riseup.net

(Error code: sec_error_unknown_issuer)

- This could be a problem with the server's configuration, or it could be someone trying to impersonate the server.
- If you have connected to this server successfully in the past, the error may be temporary, and you can try again later.

Or you can add an exception...
Now: An example from IE

There is a problem with this website’s security certificate.

The security certificate presented by this website was not issued by a trusted certificate authority.
The security certificate presented by this website was issued for a different website’s address.

Security certificate problems may indicate an attempt to fool you or intercept any data you send to the server.

We recommend that you close this webpage and do not continue to this website.

- Click here to close this webpage.
- Continue to this website (not recommended).
- More Information
Conclusions

If we trigger the negative feedback, we're screwed.
If we fail to trigger the positive feedback, it's not so bad.
How is SSL used?
Nobody types https://
(or http:// for that matter)
People generally encounter SSL in only two ways:

Clicking on links.
Through 302's.
Which means that people only encounter SSL through HTTP...
First cut: A different kind of MITM

Normally we attack the SSL connection...
First cut: A different kind of MITM

What if we attacked the HTTP connection instead...
Remember:
SSL is normally encountered in one of two ways.

By clicking on links.
Through 302 redirects.

We can attack both of those points through a HTTP MITM.
A First Cut Recipe: sslstrip

Watch HTTP traffic go by.
Switch `<a href="https://...">` to `<a href="http://...">` and keep a map of what's changed.
Switch Location: https://... to Location: http://... and keep a map of what's changed.
A First Cut Recipe: sslstrip

Watch HTTP traffic go by. When we see an HTTP request for a URL that we've stripped, proxy that out as HTTPS to the server. Watch the HTTPS traffic go by, log everything if we want, and keep a map of the relative links, CSS links, and JavaScript links that go by.
A First Cut Recipe: sslstrip

The server never knows the difference. Everything looks secure on their end. The client doesn't display any of the disastrous warnings that we want to avoid. We see all the traffic.
How does it look?
Secure Site

Gmail: Email from Google - Mozilla Firefox

Welcome to Gmail

A Google approach to email.

Gmail is a new kind of webmail, built on the idea that email can be more intuitive, efficient, and useful. And maybe even fun. After all, Gmail has:

Less spam
Keep unwanted messages out of your inbox with Google's innovative technology.

Mobile access
Read Gmail on your mobile phone by pointing your phone's web browser to http://gmail.com/app. Learn more

Lots of space
Over 7,290,461 megabytes (and counting) of free storage so you'll never need to delete another message.

Sign in to Gmail with your
Google Account

Sign up for Gmail

©2009 Google · Gmail for Organizations · Gmail Blog · Terms · Help
Secure Site

A Google approach to email.

Gmail is a new kind of webmail, built on the idea that email can be more intuitive, efficient, and useful. And maybe even fun. After all, Gmail has:

- **Less spam**
  Keep unwanted messages out of your inbox with Google’s innovative technology.

- **Mobile access**
  Read Gmail on your mobile phone by pointing your phone’s web browser to [http://gmail.com/app](http://gmail.com/app). Learn more.

- **Lots of space**
  Over 7290.462157 megabytes (and counting) of free storage so you’ll never need to delete another message.

Sign in to Gmail with your Google Account

Username:

Password:

- Remember me on this computer.

Sign in

I cannot access my account

Sign up for Gmail

About Gmail · New features!
Secure Site

A Google approach to email.

Gmail is a new kind of webmail, built on the idea that email can be more intuitive, efficient, and useful. And maybe even fun. After all, Gmail has:

- **Less spam**
  Keep unwanted messages out of your inbox with Google's innovative technology.

- **Mobile access**
  Read Gmail on your mobile phone by pointing your phone's web browser to [http://gmail.com/app](http://gmail.com/app).
  Learn more

- **Lots of space**
  Over 7256.652889 megabytes (and counting) of free storage so you'll never need to delete another message.
A Google approach to email.

Gmail is a new kind of webmail, built on the idea that email can be more intuitive, efficient, and useful. And maybe even fun. After all, Gmail has:

Less spam
Keep unwanted messages out of your inbox with Google's innovative technology.

Mobile access
Read Gmail on your mobile phone by pointing your phone's web browser to http://gmail.com/app. Learn more

Lots of space
Over 7256.653389 megabytes (and counting) of free storage so you'll never need to delete another message.
What else can we do?

We've managed to avoid the negative feedback, but some positive feedback would be good too. People seem to like the little lock icon thing, so it'd be nice if we could get that in there too.
A 1.5 Cut: sslstrip

A new trick:

Let's do everything the same, but now watch out for favicon requests as well.
If we see a favicon request for a URL that we've stripped, we'll send back a favicon of our choosing instead.
What should our favicon be? You guessed it:
Once again, a secure site:

A Google approach to email.

Gmail is a new kind of webmail, built on the idea that email can be more intuitive, efficient, and useful. And maybe even fun. After all, Gmail has:

- **Less spam**
  Keep unwanted messages out of your inbox with Google's innovative technology.

- **Mobile access**
  Read Gmail on your mobile phone by pointing your phone's web browser to [http://gmail.com/app](http://gmail.com/app). [Learn more](http://gmail.com/app).

- **Lots of space**
  Over 7,290,461,681 megabytes (and counting) of free storage so you'll never need to delete another message.

Sign in to Gmail with your Google Account

- **Username:**
- **Password:**

[Sign in](#)

[Sign up for Gmail](#)

[About Gmail](#)  [New features!](#)

©2009 Google · [Gmail for Organizations](#) · [Gmail Blog](#) · [Terms](#) · [Help](#)
Once again, a secure site:
We're doing pretty good.
We've avoided the negative feedback of death.
We can do a subtle MITM via HTTP.
And if we want we can throw in a little lock icon.
Some sites provide no visible difference.
Some sites provide no visible difference.
The sites themselves confuse us.
The sites themselves confuse us.
A Few Gotchas

Content encodings that are difficult to parse (compress, gzip, etc...)

Secure cookies won't get sent over HTTP that's been stripped of SSL.

Cached pages that don't give us a chance to swap out their links.
A Few Gotchas

Content encodings that are difficult to parse (compress, gzip, etc...)

Secure cookies won't get sent over HTTP that's been stripped of SSL.

Cached pages that don't give us a chance to swap out their links.

A Simple Solution

Strip all that stuff too.

Kill the secure bit on Set-Cookie statements, strip the content encodings we don't like from client requests, and strip if-modified-since headers too.
Another problem: sessions

The most interesting stuff to log are POSTs that would have been sent via SSL. Particularly, usernames/passwords. Sessions often cause us to miss the login step, which is unfortunate. Sure, we can get the session cookie, but that's small change.
So let's strip sessions too.

Request
302 for the same URL, but with Set-Cookie: headers that expire all the cookies we got from the request.

Request Again (Sans-Cookies)

sslstrip
And a little less sketchy...

Sessions expire, and it's not always clear when or why, but they don't usually expire right in the middle of an active session. So what we do now:

When we start a MITM against a network, strip all the traffic immediately, but don't touch the cookies for 5 min (or some specified length of time).

As the cookies go by, make note of the active sessions.

After the time is up, start killing sessions, but only new sessions that we haven't seen before. These should be the “long running” sessions that won't be seen as suspicious should they disappear.
Some Results Of This Trick?

- login.yahoo.com 114
- Gmail 50
- ticketmaster.com 42
- rapidshare.com 14
- Hotmail 13
- paypal.com 9
- linkedin.com 9
- facebook.com 3
In 24 Hours

117 email accounts.
16 credit card numbers.
7 paypal logins.
Over 300 other miscellaneous secure logins.
Number of people that balked.

0
Where can we go from here?
Combining this technique with homograph attacks.

Standard homograph attack:

Sometimes the glyphs of different characters look alike. PayPal.com looks like paypal.com but is really paypai.com

Made more interesting by IDN. It became possible to register a domain with characters that appear identical to the glyphs of characters in the Latin character set.

In 2005, Eric Johanson registered p&#1072;ypal.com, which uses the Cyrillic 'a' look-alike character and displays as paypal.com
Combining this technique with homograph attacks.

What I don't like about the standard attack:

The attack vector has to be targeted. By registering \( p\&\#1072;\)ypal.com, all we can attack is paypal.com

Phishing is really just too much work. It'd be nicer if we could just MITM a network and get whatever people are doing.

The IDN stuff has been fixed. For TLDs like .com, Firefox renders the IDN characters as punycode both in the URL bar and the status bar.
p&ypal.com today

This website is hosted by 3ric Johanson. More information can be found Here. Also, if you happen to work for paypal, please get in touch with me. I'd like to give you your trademarked domain name back... please...
So how can we reinvent this to attack SSL?

We can't use .com or any TLD that Firefox will render into punycode.

We want something that we can generalize, not just a simple substitution for some particular character in a domain.

So, what's in most URLs? . / & ?
one trick

Register a domain like ijjk.cn
Get a domain-validated SSL wildcard cert for *.ijjk.cn
Use IDN-valid characters that look very similar to '/' and '?' to create false URLs.
MITM HTTP and swap out the HTTPS links as usual.
But this time, instead of just stripping the HTTPS links, we swap them out for our own look-alikes.
one trick

https://www.gmail.com/accounts/ServiceLogin becomes
https://www.gmail.com/accounts/ServiceLogin?!f.ijjk.cn

The latter does not display as punycode in the status bar or the URL bar.

When resolved, it becomes www.google.xn--comaccountsservicelogin-5j9pia.f.ijjk.cn

When we MITM these connections, we do SSL on both ends, but are able to present our own valid *.ijjk.cn cert to the client.
Here We Go

Request
302 for the same URL, but with Set-Cookie: headers that expire all the cookies we got from the request.
Request Again (Sans-Cookies)
Proxy HTTP back, and swap out all the HTTPS links for our own look-alike HTTPS links.
SSL request for a look-alike domain that we control.
Proxy data back from the actual domain.
An Example
Nice thing about this...

Happens in real-time.

Generalized:

Targets whatever secure sites people are browsing to at any moment.

Doesn't require multiple certificates or restricting ourselves to popular sites.

Once we get a secure POST, we can switch them back to a normal traffic stream.
Lessons...

Lots of times the security of HTTPS comes down to the security of HTTP, and HTTP is not secure. If we want to avoid the dialogs of death, start with HTTP not HTTPS.

Once we've got control of that, we can do all kinds of stuff to re-introduce the positive indicators people might miss.
Other tricks...
sslstrip

http://www.thoughtcrime.org