

# MORE TRICKS FOR DEFEATING SSL IN PRACTICE

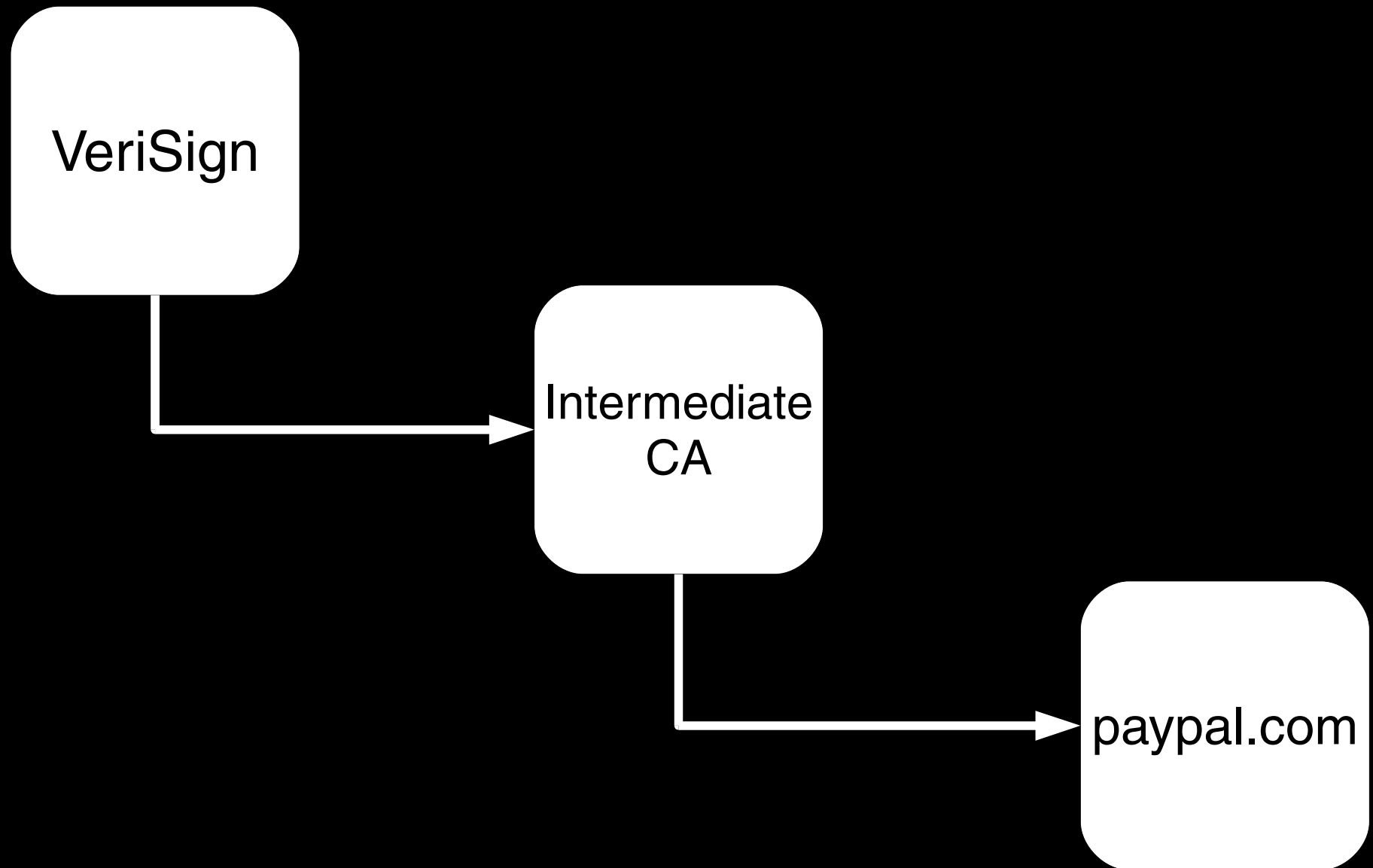
Moxie Marlinspike  
[moxie@thoughtcrime.org](mailto:moxie@thoughtcrime.org)



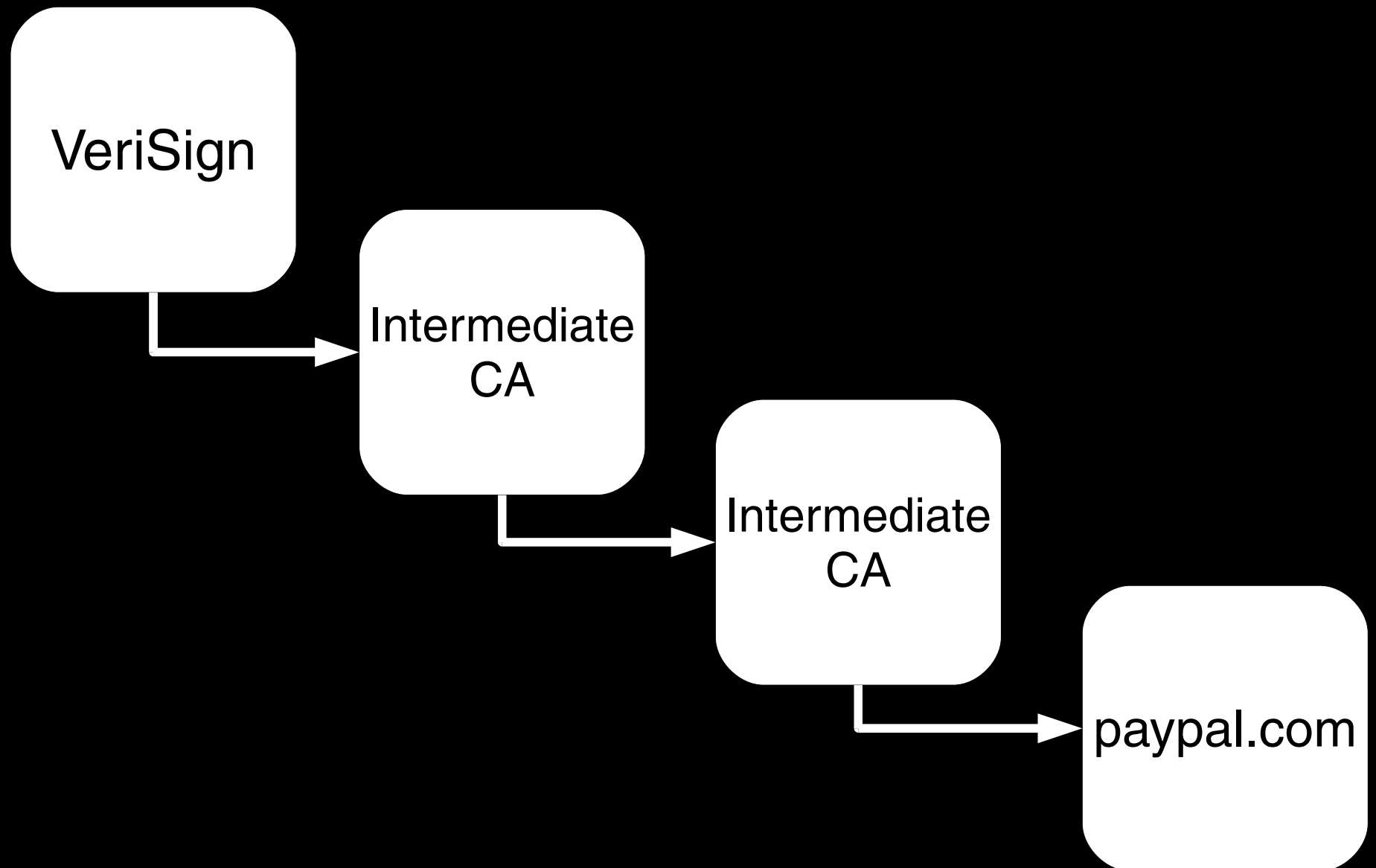
ONCE AGAIN, THE BACK STORY...

IN THE PAST, I'VE TALKED ABOUT  
BASICCONSTRAINTS...

# CERTIFICATE CHAINING



# CERTIFICATE CHAINING



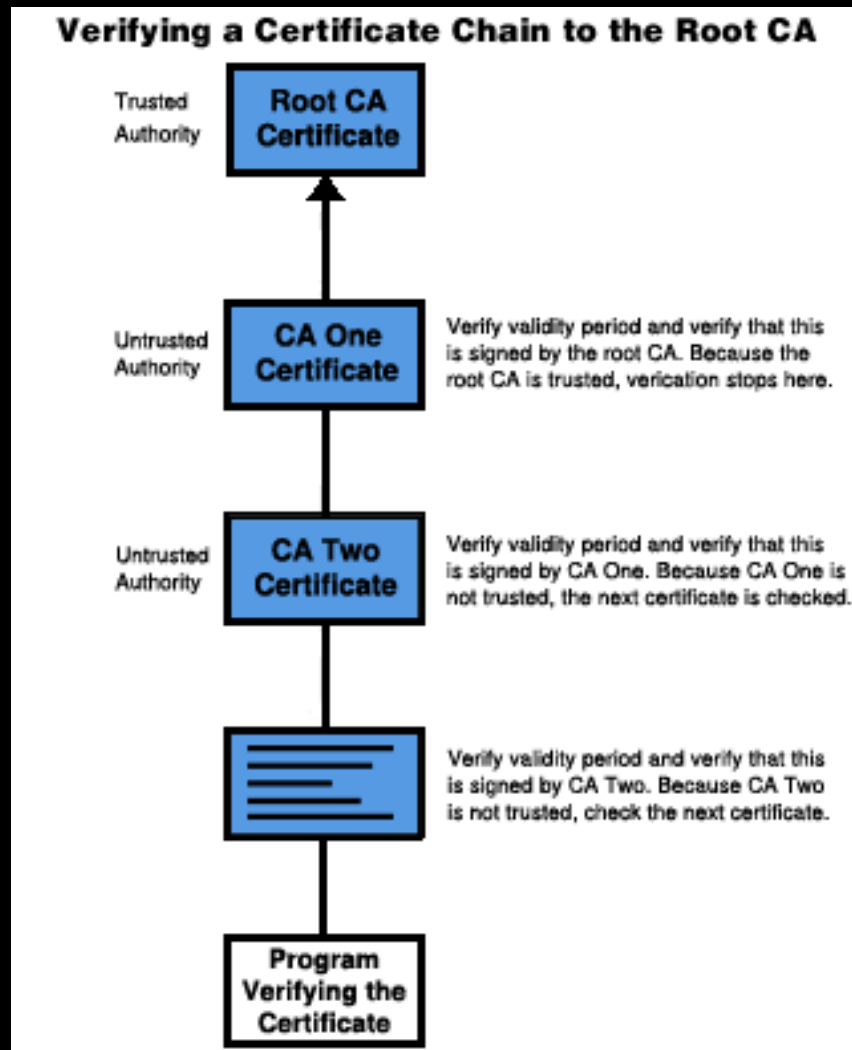
HOW DO WE VERIFY THESE THINGS?

# WHAT THEY SAY:

- Verify that the name of the leaf node is the same as the site you're connecting to.
- Verify that the leaf certificate has not expired.
- Check the signature.
- If the signing CA is in our list of trusted root CAs, stop. Otherwise, move one up the chain and repeat.

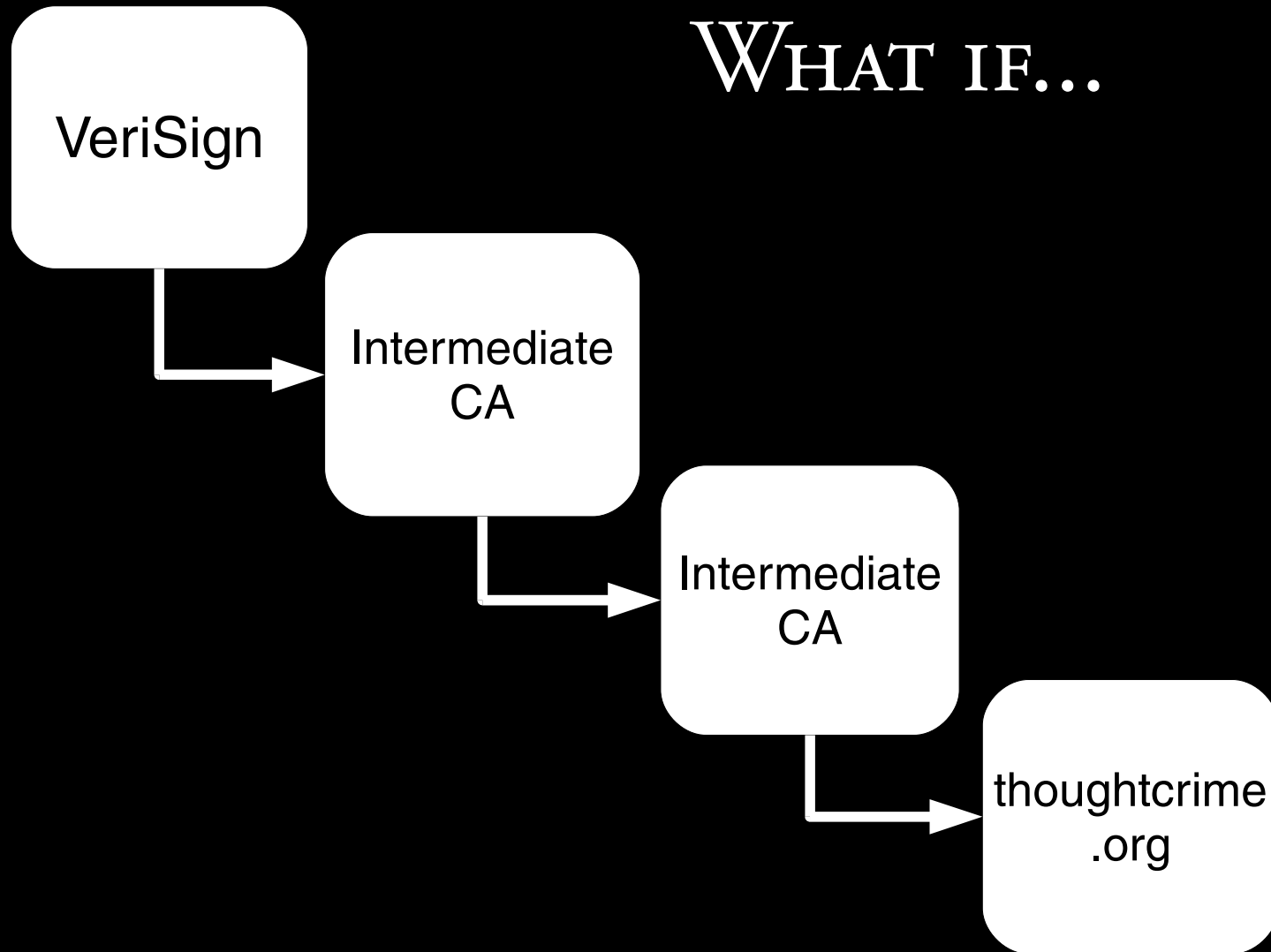


# HERE BE DRAGONS

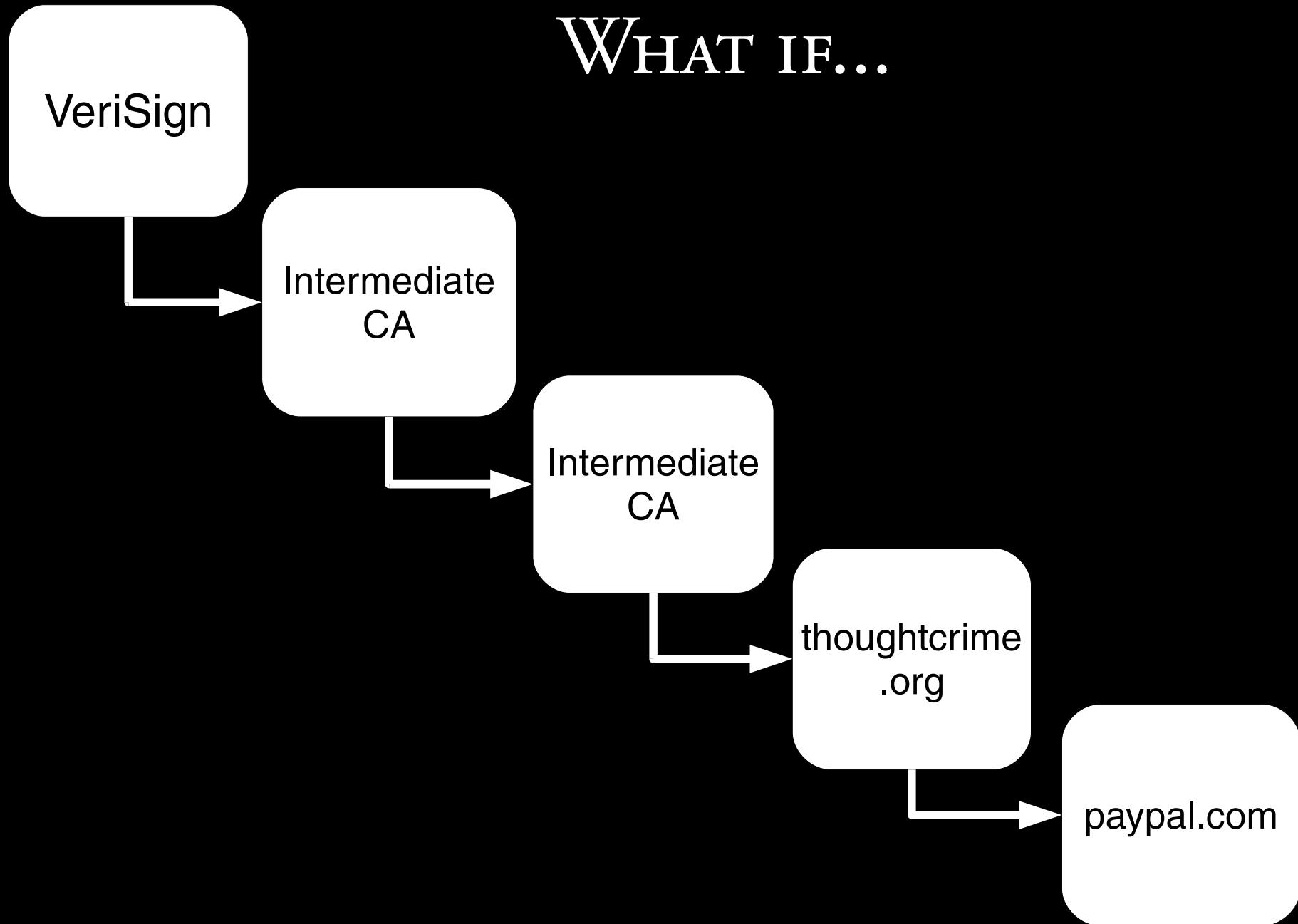


- Very tempting to use a simple recursive function.
- Everyone focuses on the signature validation.
- The result of a naïve attempt at validation is a chain that is complete, but nothing more.

# WHAT IF...



# WHAT IF...



# WHAT THEY SAY:

- Verify that the name of the leaf node is the same as the site you're connecting to.
- Verify that the leaf certificate has not expired.
- Check the signature.
- If the signing CA is in our list of trusted root CAs, 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.

```
File Edit View Terminal Tabs Help
moxie@searching: ~/Desktop/b... X moxie@searching: ~/Desktop/b... X moxie@searching: ~/Desktop/b... X
      f8:c9:0f:24:d2:c7:c2:92:0c:13:54:93:d5:9b:c7:
      0e:fa:19:a8:d5:d3:f7:ab:5d
      Exponent: 65537 (0x10001)
X509v3 extensions:
      X509v3 Key Usage: critical
      Digital Signature, Non Repudiation, Key Encipherment, Data Encip
herment
      X509v3 Subject Key Identifier:
      DF:48:EF:25:BF:D2:23:B0:F0:C2:AC:FA:5A:85:50:74:FF:F9:34:EF
      X509v3 CRL Distribution Points:
      URI:http://crl.geotrust.com/crls/globalca1.crl

      X509v3 Authority Key Identifier:
      keyid:BE:A8:A0:74:72:50:6B:44:B7:C9:23:D8:FB:A8:FF:B3:57:6B:68:6
C

      X509v3 Extended Key Usage:
      TLS Web Server Authentication, TLS Web Client Authentication
      X509v3 Basic Constraints: critical
      CA:FALSE
Signature Algorithm: sha1WithRSAEncryption
      7a:58:f9:88:14:cb:77:32:aa:83:12:de:d9:15:74:8e:34:e3:
      66:ca:bc:24:2c:28:96:54:cd:be:51:56:60:87:e3:be:c6:2e:
      86:7e:74:c1:68:01:b6:8c:07:c6:a2:0c:a4:36:ca:e1:a8:e9:
```



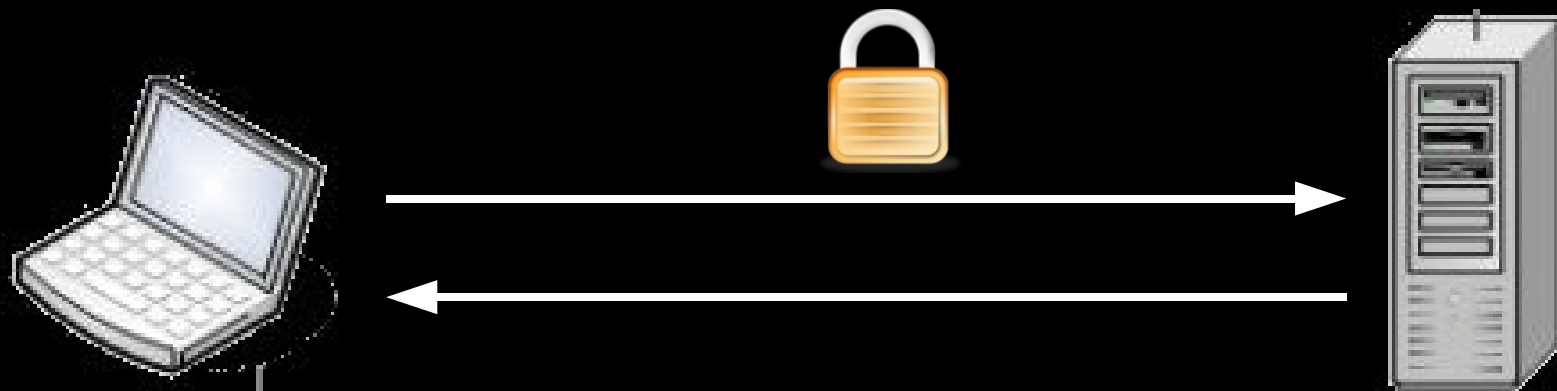
# BACK IN THE DAY

- Most CAs didn't explicitly set basicConstraints: CA=False
- Whether the field was there or not, most SSL implementations didn't bother to check it.
- *Anyone* with a valid leaf node certificate could create and sign a leaf node certificate for *any other* domain.
- When presented with a complete chain, IE, Outlook, Konqueror, OpenSSL, and others considered it valid...

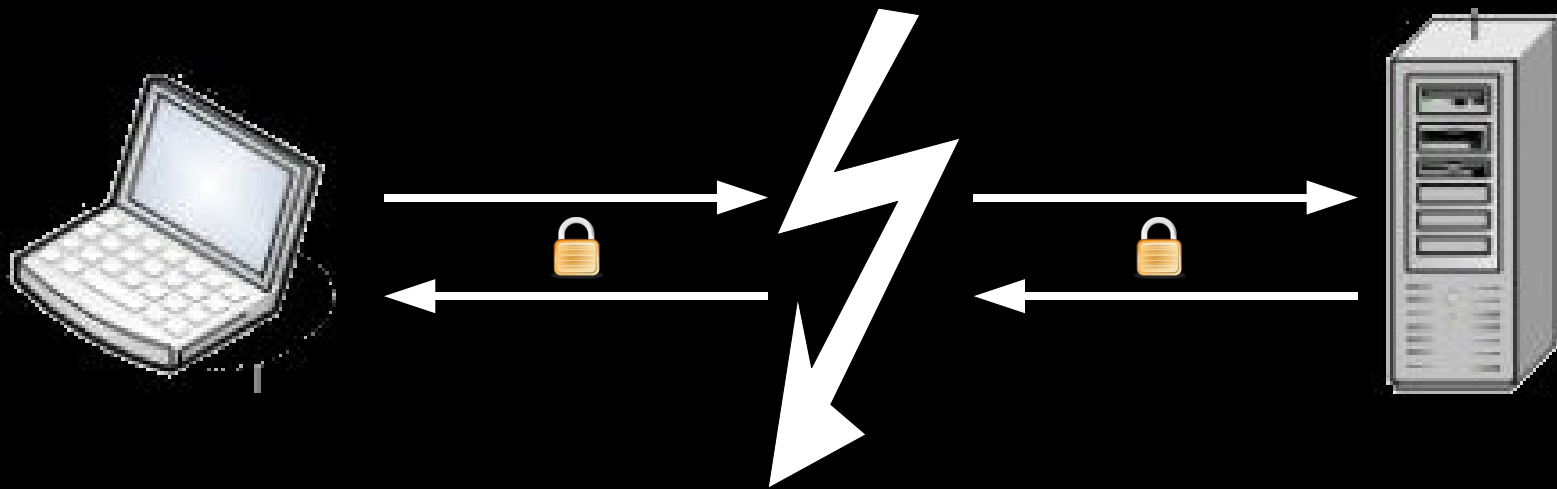
## AND THEN IN 2002...

- Microsoft did something particularly annoying, so I blew this up by publishing it.
- Microsoft claimed that it was impossible to exploit.
- So I also published the tool that exploits it.

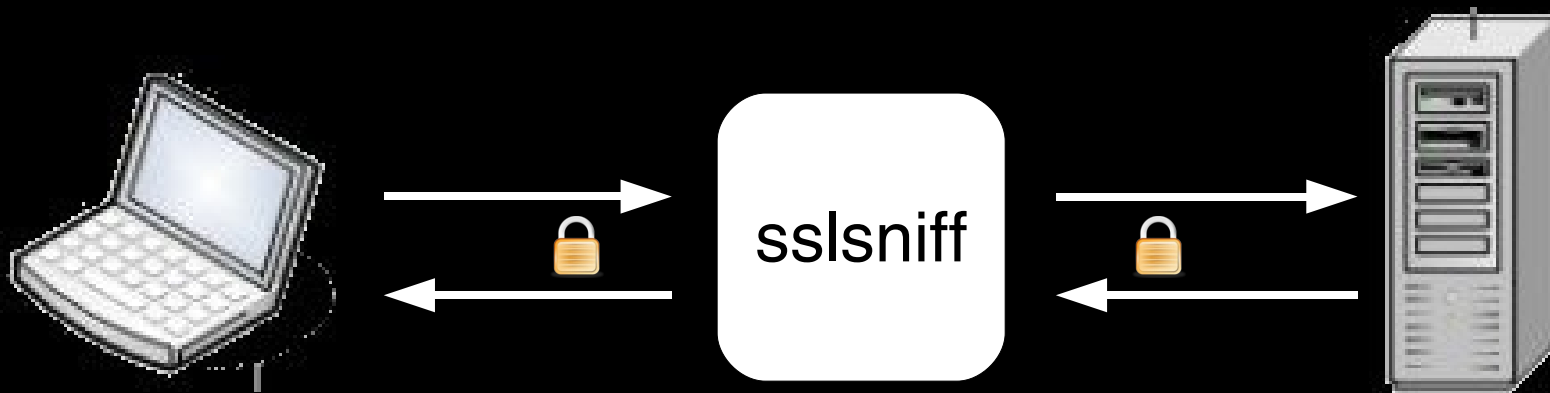
# SSLSNIFF



# SSLSNIFF



# SSLSNIFF

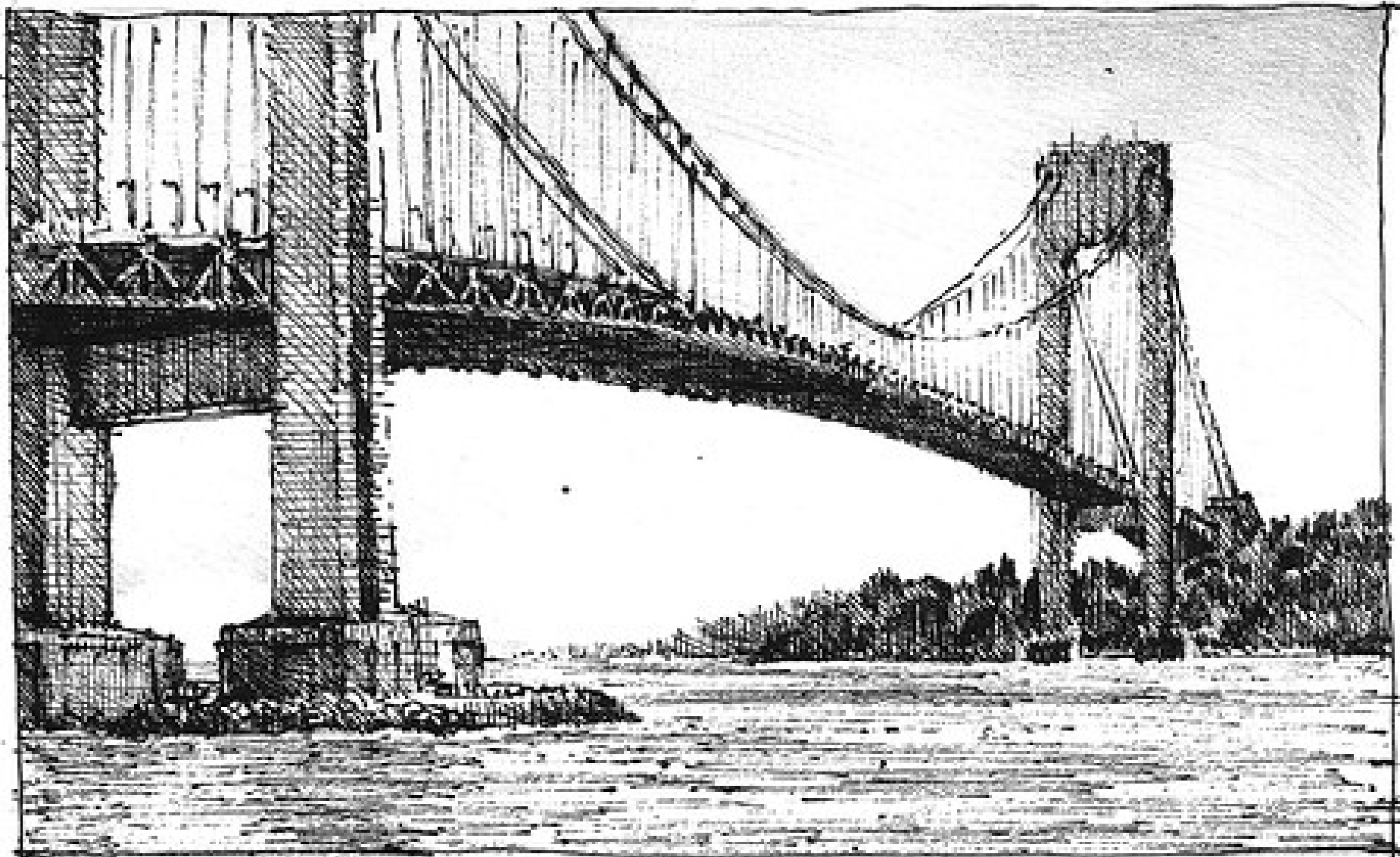


- Intercept a connection from the client side.
- Generate a certificate for the site it is connecting to.
- Sign it with any random valid leaf node certificate.
- Pass that certificate chain to the client.

- Make a normal SSL connection to the server.
- Pass data between client and server, decrypting and encrypting on each end.

LATELY, I'VE BEEN TALKING ABOUT  
SSL STRIPPING...

# BRIEF



VERRAZANO BRIDGE — BROOKLYN

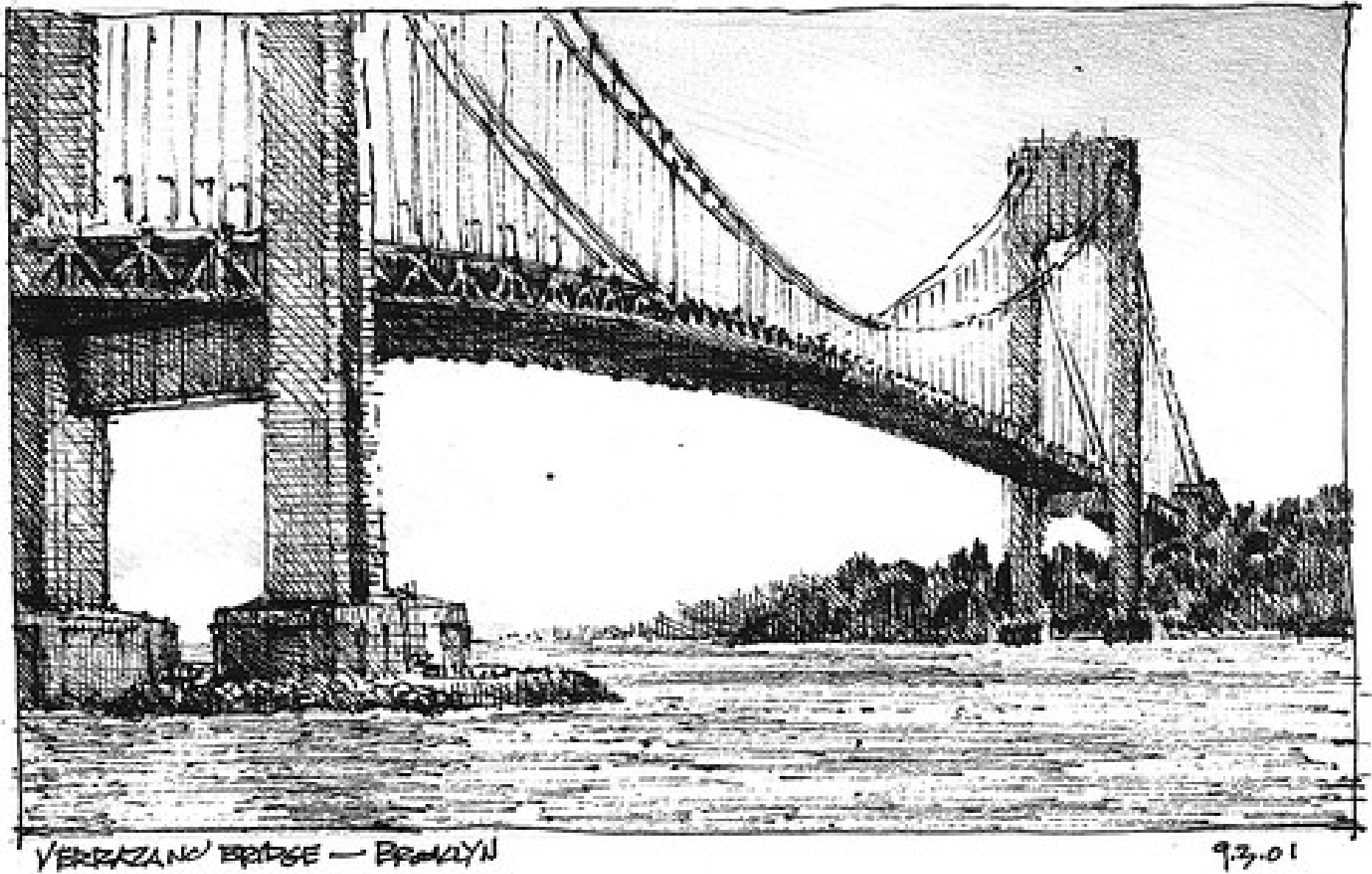
9.3.01



SSL CAN BE USEFUL, BUT HOW IT'S DEPLOYED  
*matters*

# IN THE CONTEXT OF WEB BROWSING

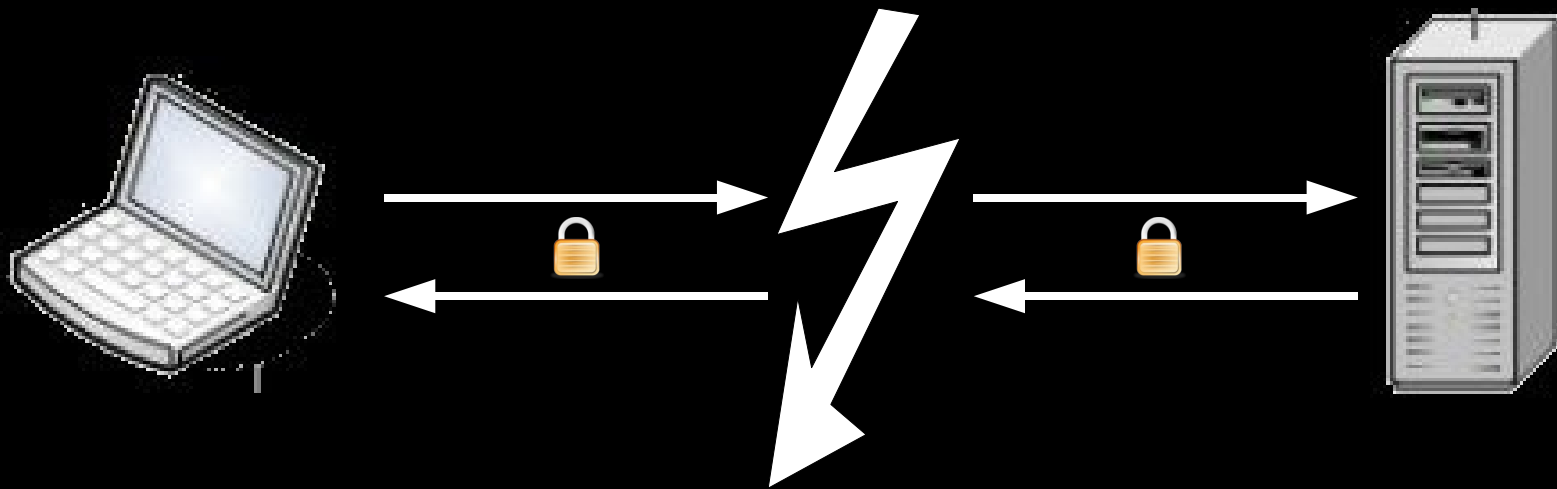
- SSL is almost never encountered directly.
- It is either encountered as a result of:
  - A 302 redirect from an HTTP URL to an HTTPS URL.
  - An HTTPS link that a user clicks on from an HTTP page.
    - (Think, “My Cart,” “Checkout,” “Login,” etc...)



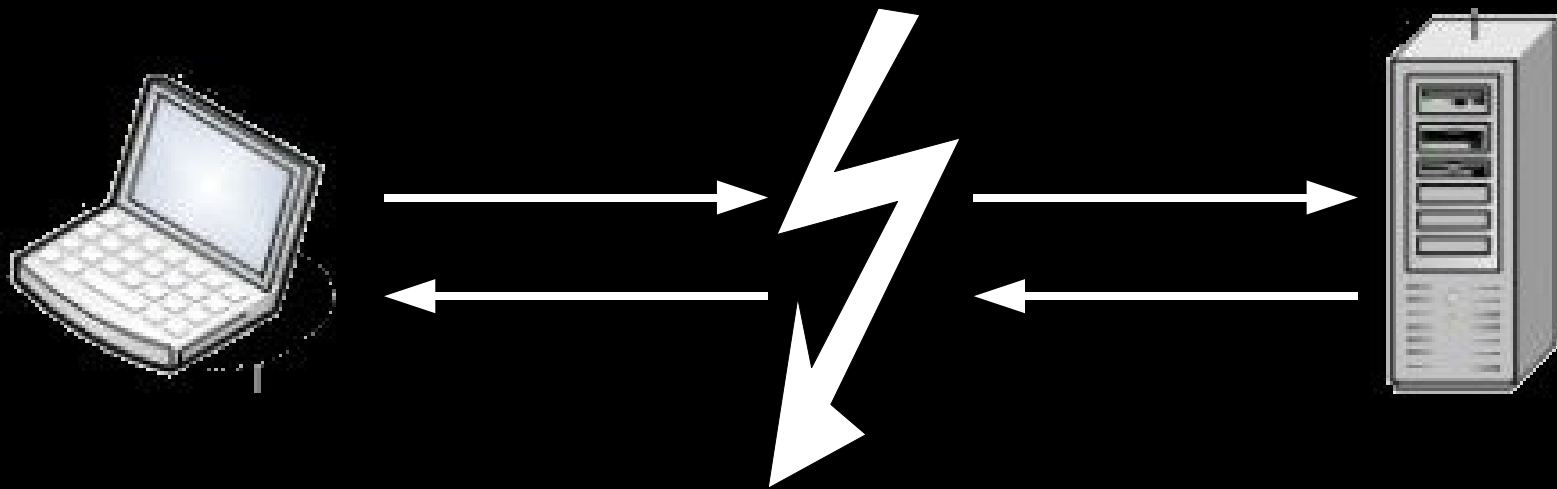


WE CAN ATTACK SSL  
BEFORE WE EVEN GET THERE

# SSLSNIFF

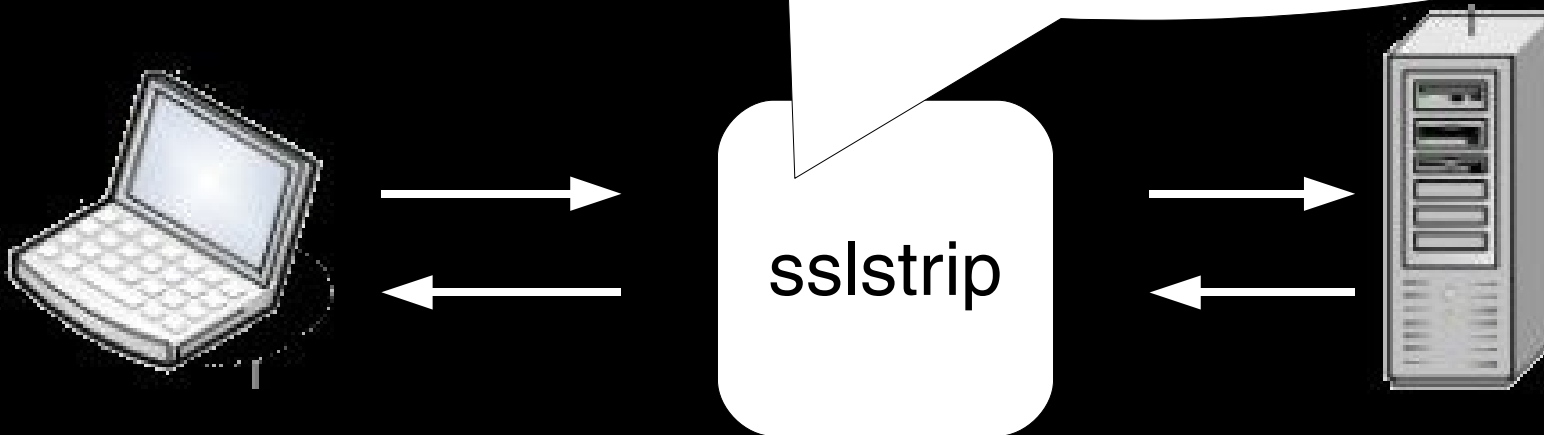


# SSLSTRIP



# SSLSTRIP

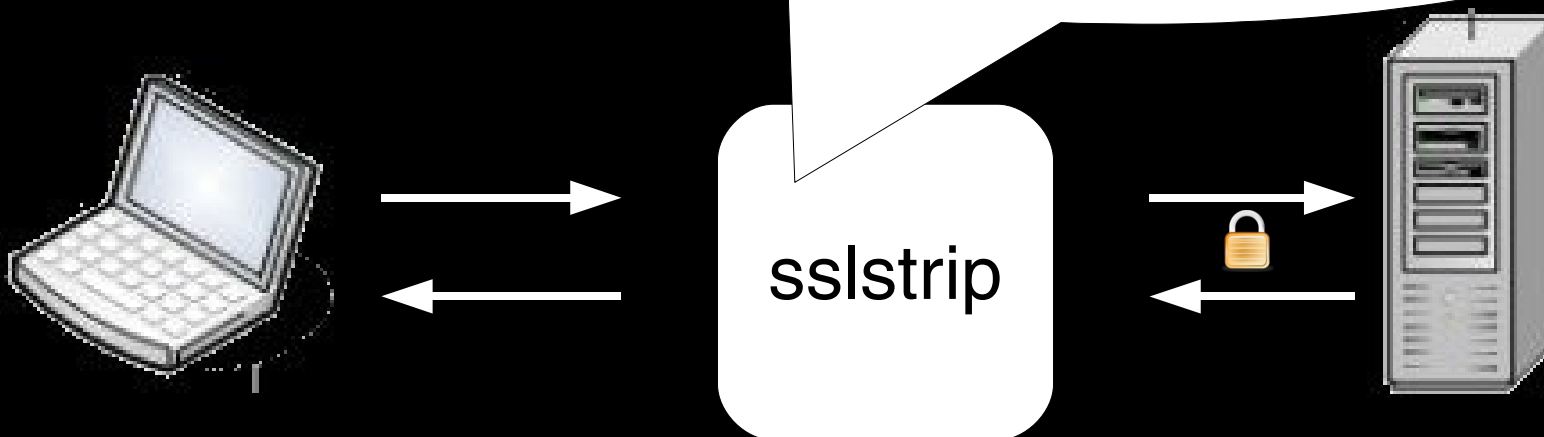
- Watch HTTP traffic go by.
- Switch `<a href="https://...">` to `<a href="http://...">` and *keep a map of what you've changed*.
- Switch Location: `https://` to Location: `http://` and *keep a map of what you've changed*.



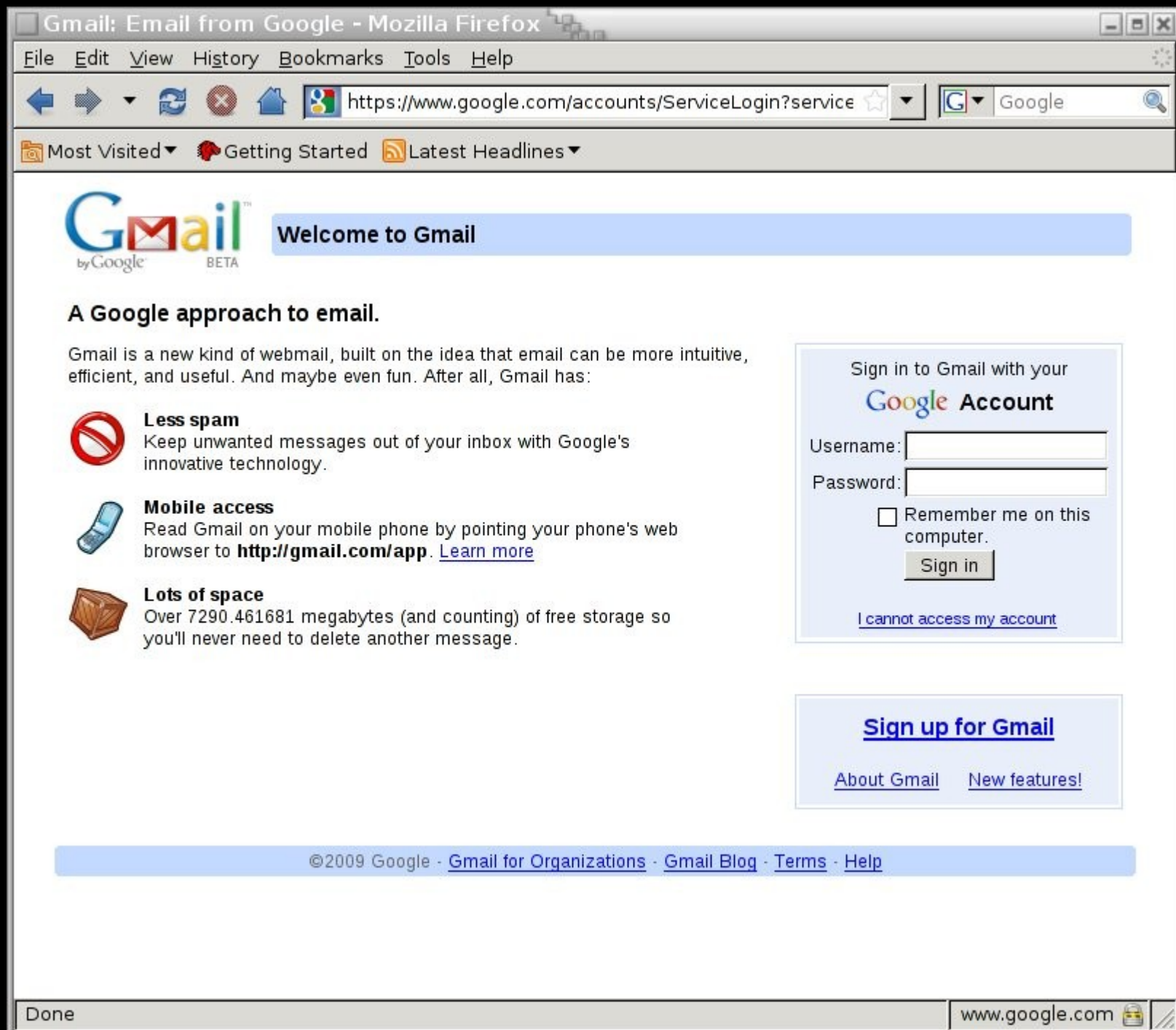


# SSLSTRIP

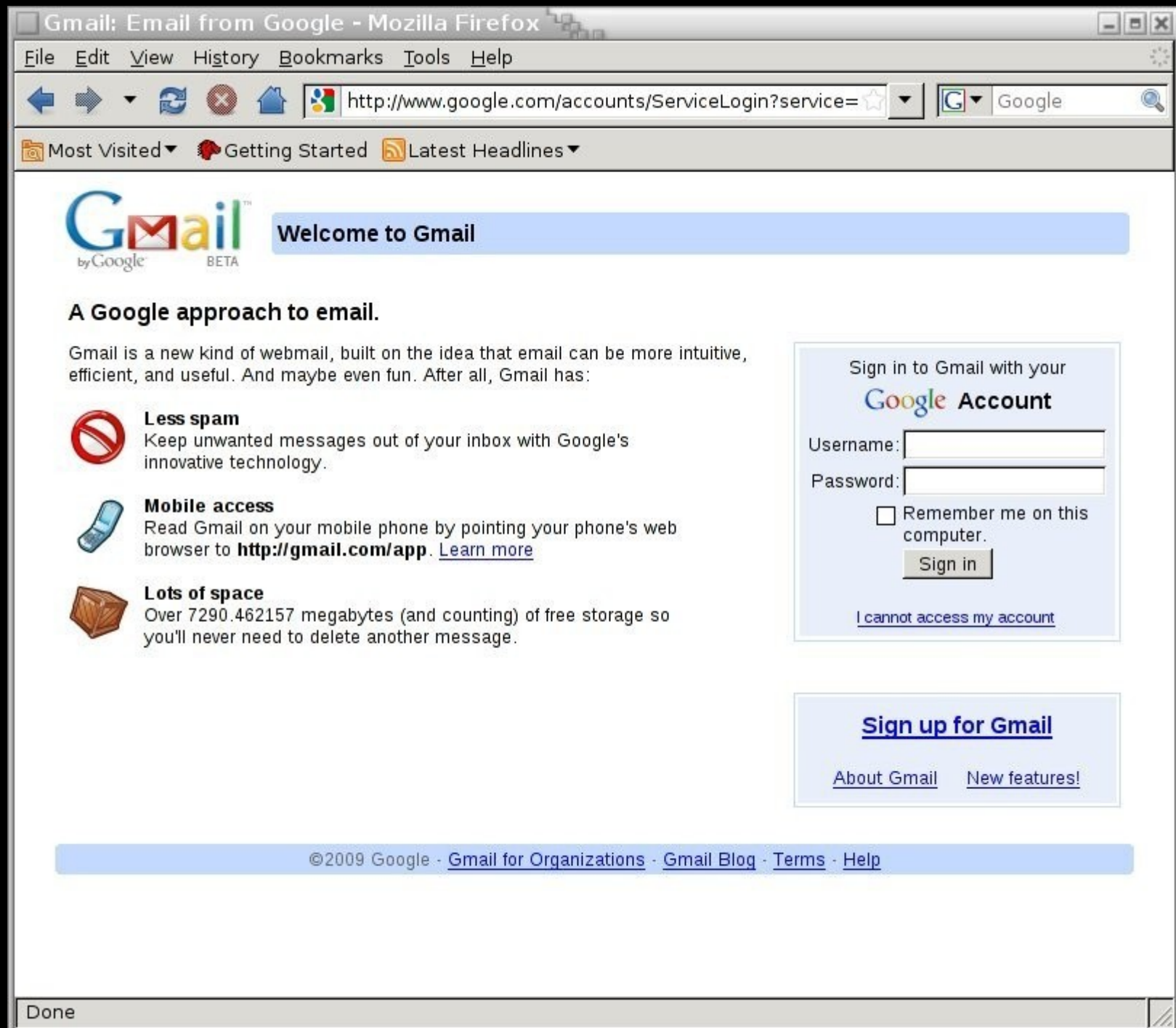
- Watch HTTP traffic go by.
- When we see an HTTP request for a URL that we've stripped, proxy that out as HTTPS to the server.
- Watch the HTTPS traffic go by, log everything that we want, and keep a map of all relative, CSS, and JS links that go by.



# How Does It Look?



# How Does It Look?



# How Does It Look?

Gmail: Email from Google

https://www.google.com/accounts/ServiceLogin?service=mail&passive=true&rm=false&co Google

Apple Yahoo! Google Maps YouTube Wikipedia News (26) Popular

**Gmail**  
by Google BETA

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 7295.652889 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.

[I cannot access my account](#)

[Sign up for Gmail](#)

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

©2009 Google - [Gmail for Organizations](#) - [Gmail Blog](#) - [Terms](#) - [Help](#)

# How Does It Look?

Gmail: Email from Google

http://www.google.com/accounts/ServiceLogin?service=mail&passive=true&rm=false&con Google

Apple Yahoo! Google Maps YouTube Wikipedia News (26) Popular

 **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 7295.653389 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.

[I cannot access my account](#)

[Sign up for Gmail](#)

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

©2009 Google - [Gmail for Organizations](#) - [Gmail Blog](#) - [Terms](#) - [Help](#)

WHERE CAN WE GO FROM HERE?

WHERE DO WE *need* TO GO FROM HERE?

# WHAT'S WITH CERTIFICATES, ANYWAYS?

## X509Certificate

Version

Serial Number

Issuer

Validity (not before X or after Y)

Subject

PublicKey

SignatureAlgorithm

Signature



# WHAT'S WITH CERTIFICATES, ANYWAYS?

## X509Certificate

Version

Serial Number

Issuer

Validity (not before X or after Y)

Subject

PublicKey

SignatureAlgorithm

Signature

# WHAT'S WITH CERTIFICATES, ANYWAYS?

## X509Certificate

Version

Serial Number

Issuer

Validity (not before X or after Y)

Subject

PublicKey

SignatureAlgorithm

Signature

# WHAT'S WITH CERTIFICATES, ANYWAYS?

## X509Certificate

Version

Serial Number

Issuer

Validity (not before X or after Y)

Subject

PublicKey

SignatureAlgorithm

Signature

# WHAT'S WITH CERTIFICATES, ANYWAYS?

## X509Certificate

Version

Serial Number

Issuer

Validity (not before X or after Y)

Subject

PublicKey

SignatureAlgorithm

Signature

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

01:2a:39:76:0d:3f:4f:c9:0b:e7:bd:2b:cf:95:2e:7a

Signature Algorithm: sha1WithRSAEncryption

Issuer: C=ZA, O=Thawte Consulting (Pty) Ltd., CN=Thawte SGC CA

Validity

Not Before: Mar 27 22:20:07 2009 GMT

Not After : Mar 27 22:20:07 2010 GMT

Subject: C=US, ST=California, L=Mountain View, O=Google Inc, CN=www.google.com

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

RSA Public Key: (1024 bit)

Modulus (1024 bit):

00:d6:b9:e1:ad:b8:61:0b:1f:4e:b6:3c:09:3d:ab:  
e8:e3:2b:b6:e8:a4:3a:78:2f:d3:51:20:22:45:95:  
d8:00:91:33:9a:a7:a2:48:ea:30:57:26:97:66:c7:  
5a:ef:f1:9b:0c:3f:e1:b9:7f:7b:c3:c7:cc:af:9c:  
d0:1f:3c:81:15:10:58:fc:06:b3:bf:bc:9c:02:b9:  
51:dc:fb:a6:b9:17:42:e6:46:e7:22:cf:6c:27:10:  
fe:54:e6:92:6c:0c:60:76:9a:ce:f8:7f:ac:b8:5a:  
08:4a:dc:b1:64:bd:a0:74:41:b2:ac:8f:86:9d:1a:  
de:58:09:fd:6c:0a:25:e0:79

Exponent: 65537 (0x10001)

X509v3 extensions:

X509v3 Extended Key Usage:

TLS Web Server Authentication, TLS Web Client Authentication, Netscape Server Gated Crypto

X509v3 CRL Distribution Points:

URI:http://crl.thawte.com/ThawteSGCCA.crl

Authority Information Access:

OCSP - URI:http://ocsp.thawte.com

CA Issuers - URI:http://www.thawte.com/repository/Thawte\_SGC\_CA.crt

X509v3 Basic Constraints: critical

CA:FALSE

Signature Algorithm: sha1WithRSAEncryption

39:b6:fb:11:bc:33:2c:c3:90:48:e3:6e:c3:9b:38:b1:42:d1:  
00:09:58:63:a0:e1:98:1c:85:f2:ef:10:1d:60:4e:51:09:62:  
f5:05:bd:9d:4f:87:6c:98:72:07:80:c3:59:48:14:e2:d6:ef:  
d0:8f:33:6a:68:31:fa:b7:bb:85:cc:f7:c7:47:7b:67:93:3c:  
c3:16:51:9b:6f:87:20:fd:67:4c:2b:ea:6a:49:db:11:d1:bd:  
d7:95:22:43:7a:06:7b:4e:f6:37:8e:a2:b9:cf:1f:a5:d2:bd:  
3b:04:97:39:b3:0f:fa:38:b5:af:55:20:88:60:93:f2:de:db:

# THE BIG THREE

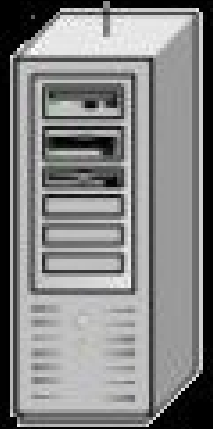
- Secrecy
- Authenticity
- Integrity



# SSL/TLS HANDSHAKE BEGINNINGS



ClientHello



ServerHello, ServerCertificate

# SSL HANDSHAKE BEGINNINGS



**Server Settings**

Server Type: IMAP Mail Server

Server Name:  Port:  Default: 99

User Name:

**Login Options**

Protocol:

Username:

Server:

## X509Certificate

Version

Serial Number

Issuer

Validity

Subject

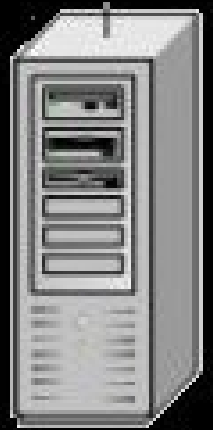
PublicKey

SignatureAlgorithm

Signature



# THE PROBLEMS FOR US BEGIN



Attacker

ClientHello



ServerHello,  
*ServerCertificate?*



LET'S START BY LOOKING BACK ONCE MORE.

IN 2000, THINGS WERE DIFFERENT.

NOTARIES!

# IDENTIFICATION!

# PHONE CALLS!

ACTUAL PEOPLE INVOLVED...

THAT IS A BYGONE ERA



# THESE DAYS IT'S ALL ABOUT: ONLINE DOMAIN VALIDATION

thawte is a leading global certificate authority of SSL certificates, extended validation ssl (EV SSL) and code signing digit

File Edit View History Bookmarks Tools Help

http://www.thawte.com/ Google

Most Visited Getting Started Latest Headlines

**thawte**™  
it's a trust thing™

worldwide sites: [Deutsch](#) [Español](#) [Français](#) [Italiano](#)

quick login:  make your selection >>  → [\[ sitemap \]](#)

[Home](#) [Products](#) [Partners](#) [Buy](#) [Renew](#) [Trials](#) [Guides](#) [Support](#)  
[Contact Us](#)

**BUY SSL CERTIFICATES**

**RENEW SSL CERTIFICATES**

hundreds of

| BUY SSL CERTIFICATES                                                                                                                                                                                                                                | DO YOU NEED TO?                                                                                                                                                                                                  | FIND OUT MORE                                                                                                                                                                                                                                                                                                 | SOLUTIONS FOR                                                                                                                                                                                                     |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Buy</b> SSL Certificates</p> <p><b>Buy</b> EV SSL Certificates</p> <p><b>Buy</b> Code Signing</p> <p><b>Renew</b> SSL Certificates</p> <ul style="list-style-type: none"> <li>Check Order Status</li> <li>Reissue SSL Certificates</li> </ul> | <ul style="list-style-type: none"> <li>Secure a Web Server</li> <li>Manage Multiple Certificates</li> <li>Secure Your Code</li> <li>Secure Your E-mail</li> <li>Offer the Highest Possible Encryption</li> </ul> | <ul style="list-style-type: none"> <li><b>Extended Validation SSL</b></li> <li><b>thawte</b> Secures IDNs</li> <li>When is a Site Really Secure</li> <li>Register for product updates</li> <li>Facts on SSL</li> <li>Key and CSR Generation</li> <li>Check your CSR</li> <li>Submit Enquiry Online</li> </ul> | <ul style="list-style-type: none"> <li>Merchants</li> <li>Hosting Companies</li> <li>Resellers</li> <li>Registrars</li> <li>Enterprises</li> <li>Educational Institutions</li> <li>Technology Partners</li> </ul> |

**FREE INTERNET SECURITY GUIDES**

|                                                                                                                                                           |                                                                                                                                                            |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>FREE</b> Secure SSL data transfer online</p> <p><b>FREE</b> Secure your Apache or IIS server</p> <p><b>FREE</b> Step-up SSL encryption with SGC</p> | <p><b>FREE</b> Sign code for secure download</p> <p><b>FREE</b> Manage multiple SSL certificates</p> <p><b>FREE</b> Browse Internet security guides...</p> |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|

**thawte SITE SEAL**

- thawte** Secured Seal
- About SSL Certificates

**Secured by thawte**  
2009-06-16

**BOOST customer confidence with NEW Extended**

**21 day free trial**

**HAVE A QUESTION?**

Done

SSL123 Digital Certificates - thawte - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Thawte Inc (US) https://www.thawte.com/process/retail/new\_ssl123?language= Google

Most Visited Getting Started Latest Headlines

**new csr required**

Please copy and paste your Certificate Signing Request (CSR) into the space below.

Paste your new CSR here. Include the full BEGIN and END lines, with dashes, as generated by your software.

```

-----BEGIN CERTIFICATE REQUEST-----
MIIBvTCCASYCAQAwTELMAkGA1UEBhMCQ0ExDzANBgNVBA
1ZWJlYzERMA8GA1UEBxMITW9udHJlYWwzGDAWBgNVBAoTD1
WFRSQUNUT1IqSU5DLjETMBEGA1UECxMKWFhYVVFjBQ1RPUj
MBkGA1UEAxMsd3d3Ln4eHRyYWN0b3luY29tMIGfMA0GCSqC
DQEBAAQUAA4GNADCBiQKBgQDknwnonVtWVq7aCyMYFM6zrTY
BD7Mjgx
FpQaZU5v/J5TEZnGC0eSj6+chv5mvJYtwZSPbadiTIL9hZ
HEh+1G1+wCkPOFDse]
AqazSL6Tslon2UEwsu83Th0cn2nfmvhsCB

```

[click for csr example](#)
[read more about a csr?](#)
[test your csr](#)

**intranet certificate**

If the certificate you are requesting is strictly for an **Intranet** server that will not be publicly accessible from the **Internet**, please click off the **Intranet cert** option below

For Intranet use only?: ☐ yes ☒ no

**additional licenses**

Additional licenses are required to secure multiple servers with one certificate. This option should only be used if all of your servers have the same common name and have the same software installed on each server to be licensed. **Click here for more information.**

If you wish to use this certificate on more than one server, you will have to buy

**information**

Customers registered in the United States should ensure that the state name included in the CSR is written out in full. For example: California, Massachusetts etc. Please do not make use of abbreviated state names, or state codes.

NOTE: Please follow the **thawte Key Generation Guidelines** on our support site to generate your CSR.

**information**

- You will be able to add additional licenses to this certificate during its lifecycle.
- You will not be able to remove licenses from a certificate once allocated.

Done

www.thawte.com

# PKCS #10

## CertificateRequest

Version

Subject

PublicKey

Attributes

# PKCS #10

CertificateRequest

Version

Subject

PublicKey

Attributes

# PKCS #10

CertificateRequest

Version

Subject

PublicKey

Attributes



[www.bankofamerica.com](http://www.bankofamerica.com)

# PKCS #10

CertificateRequest

Version

Subject

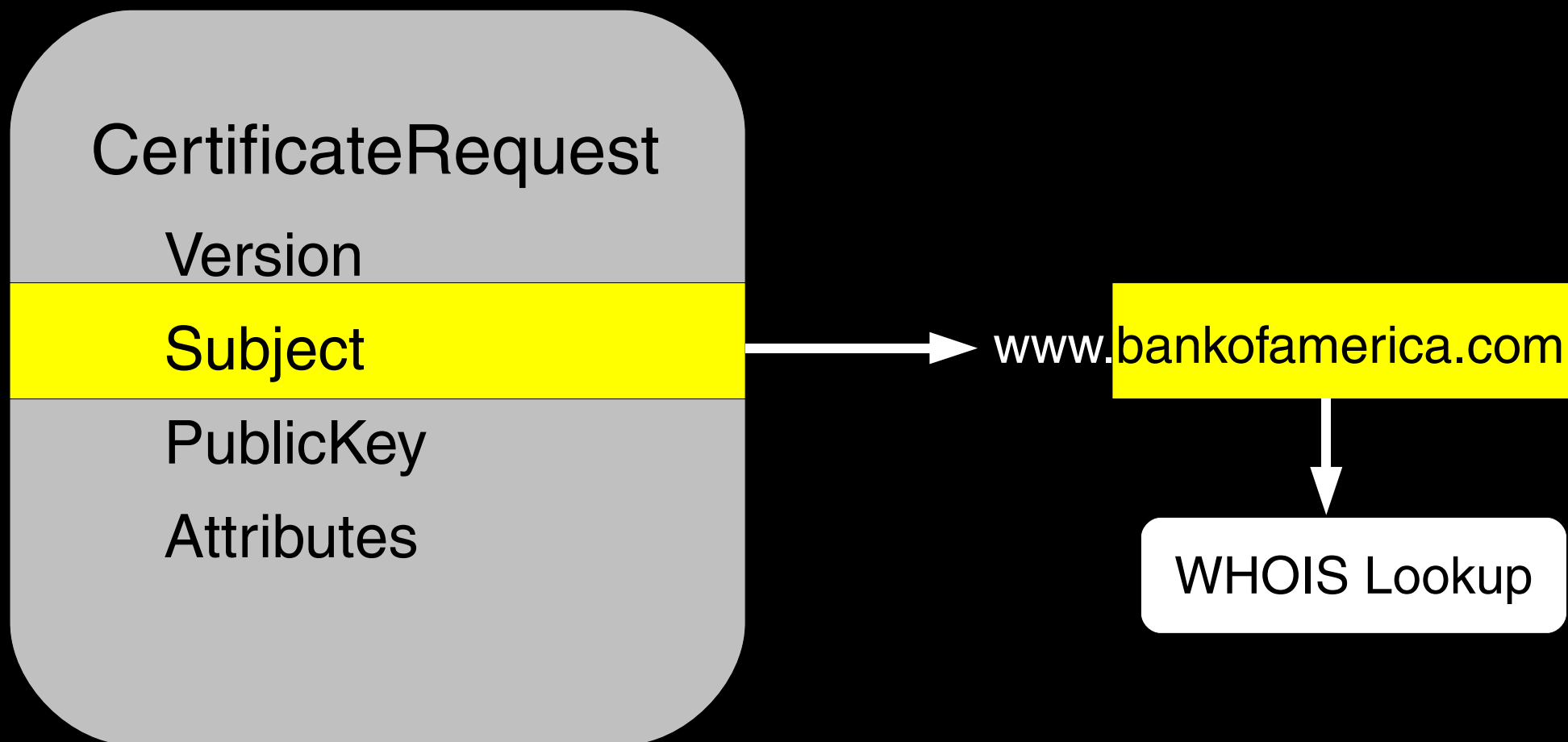
PublicKey

Attributes



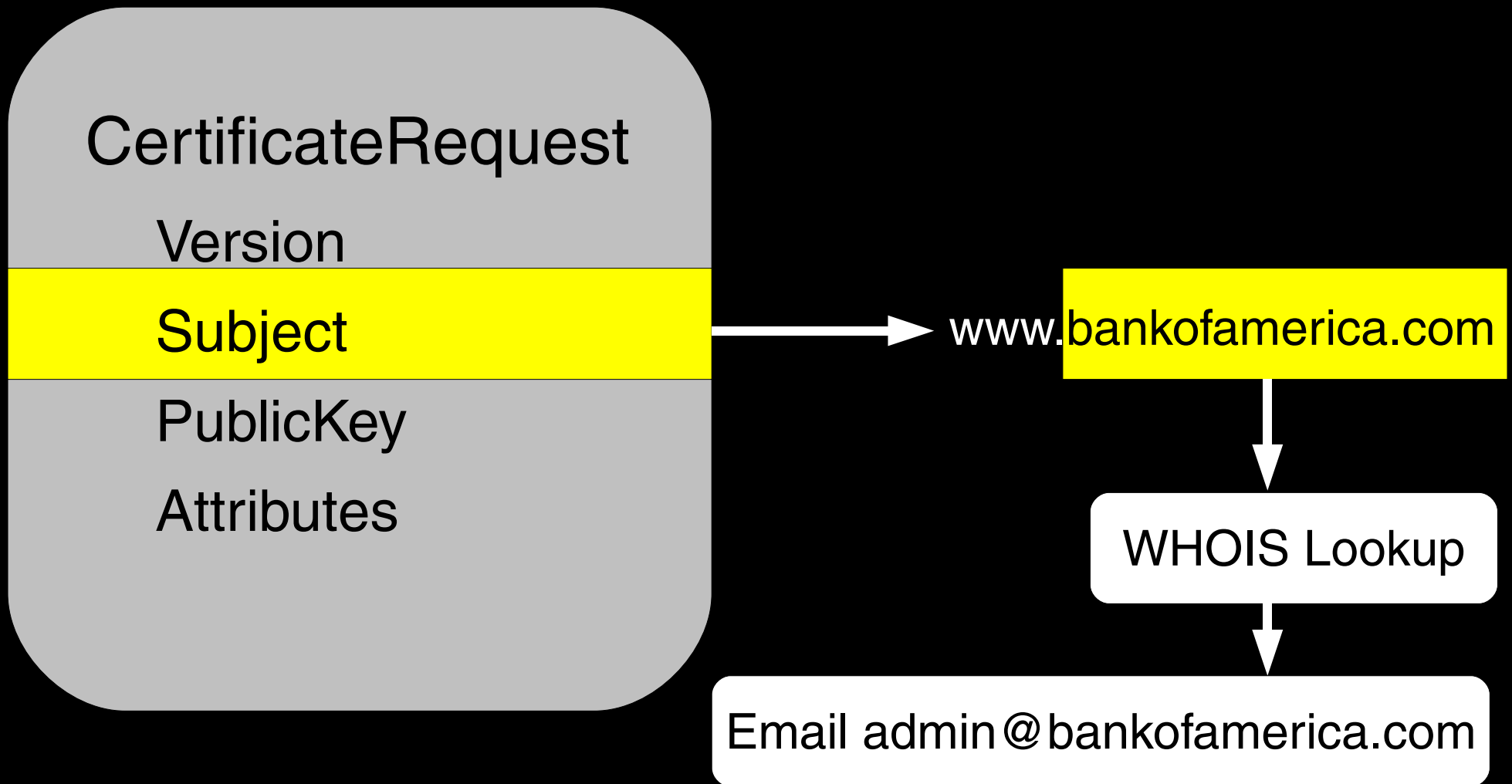
www.bankofamerica.com

# PKCS #10





# PKCS #10



# PKCS #10

CertificateRequest

Version

Subject

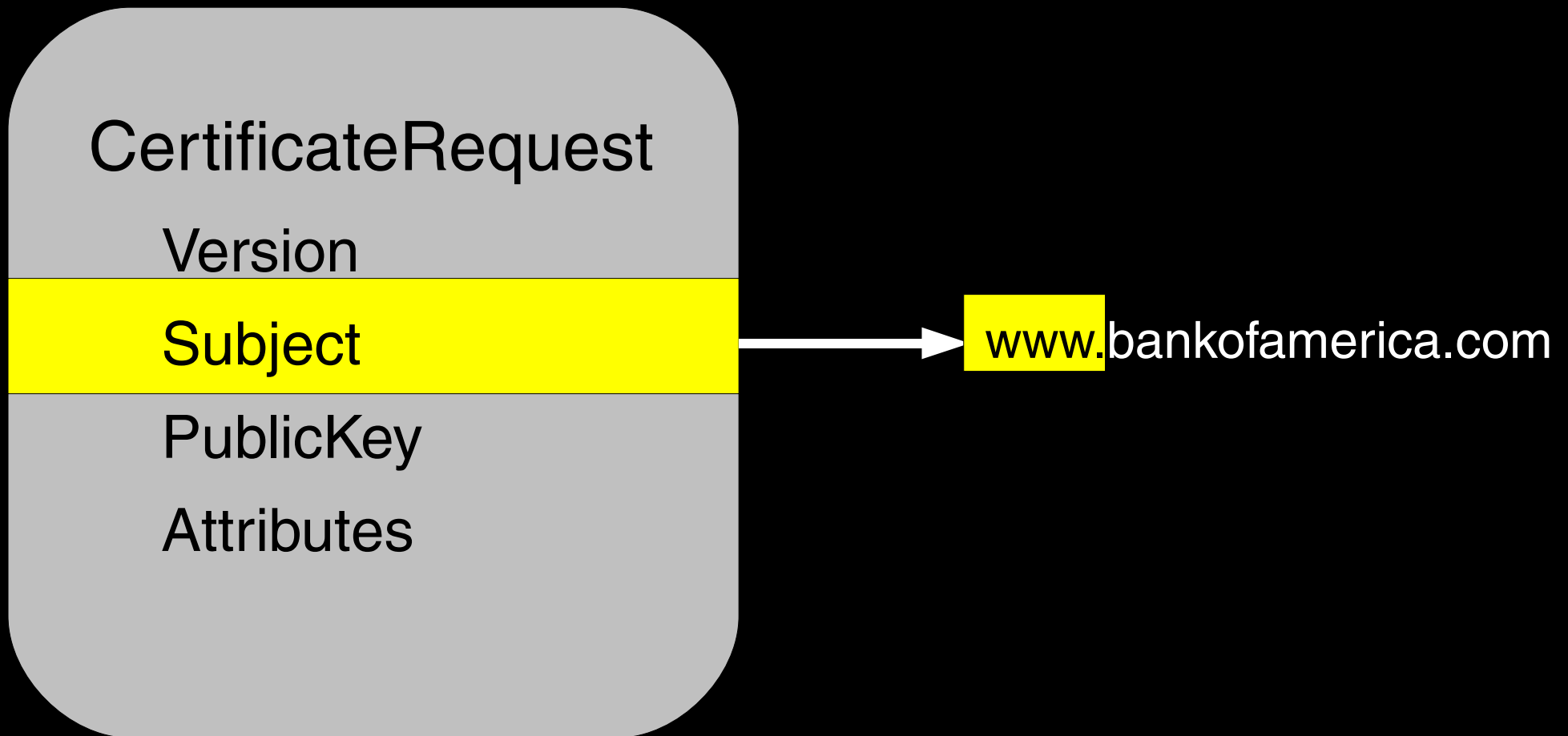
PublicKey

Attributes

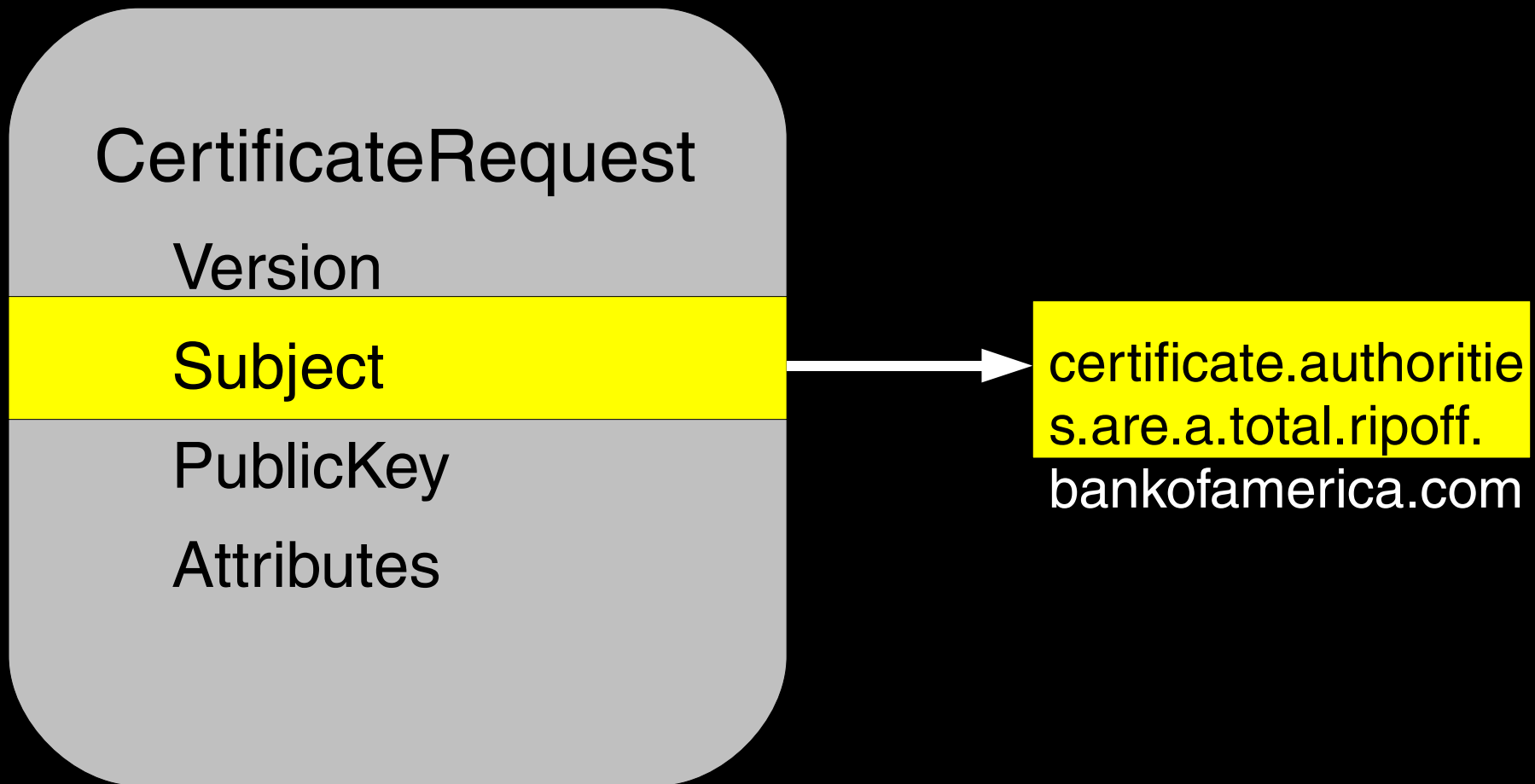


www.bankofamerica.com

# PKCS #10



# PKCS #10



# PKCS #10

CertificateRequest

Version

Subject

PublicKey

Attributes



certificate.authorities  
are a total ripoff.  
**bankofamerica.com**

# SUBJECTS

DistinguishedName

Country

State

Locale

Organization

Organizational Unit

Common Name

# SUBJECTS

DistinguishedName

Country

State

Locale

Organization

Organizational Unit

Common Name

- The X.509 standard is a total nightmare.
- Three revisions, twenty years.
- Parts of the standard have literally been “lost” and then later “found” again.

# SUBJECTS

## DistinguishedName

Country

State

Locale

Organization

Organizational Unit

Common Name

- The original vision for the DN was that each DN would fit into some global Directory Information Tree.
- In practice, the standard is weak, everyone does everything differently, and the global DIT never materialized.



# SUBJECTS

DistinguishedName

Country

State

Locale

Organization

Organizational Unit

Common Name

- “There is nothing in any of these standards that would prevent me from including a 1 gigabit MPEG movie of me playing with my cat as one of the RDN components of the DN in my certificate.”

-- Bob Jueneman on IETF-PKIX

# SUBJECTS

DistinguishedName

Country

State

Locale

Organization

Organizational Unit

Common Name



[www.bankofamerica.com](http://www.bankofamerica.com)

# CN

commonName ::=

SEQUENCE { { 2 5 4 3 }, StringType( SIZE( 1...64 ) ) }

- IA5String:
  - 0x16 – ID
  - 0x05 – Length (5 Chars)
  - 0x76, 0x61, 0x6c, 0x75, 0x65 – v, a, l, u, e

# CN ENCODING

- Essentially, the CN field is represented as a PASCAL String.

|     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|-----|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0xe | w | w | w | . | p | a | y | p | a | l | . | c | o | m |
|-----|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

- This is different from how C strings are represented.

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |    |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|----|
| w | w | w | . | p | a | y | p | a | l | . | c | o | m | \0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|----|

# PKCS #10 SUBJECT

DistinguishedName

Country

State

Locale

Organization

Organizational Unit thoughtcrime.org

Common Name

www.paypal.com

# PKCS #10 SUBJECT

Common Name



[www.thoughtcrime.org](http://www.thoughtcrime.org)

# PKCS #10 SUBJECT

Common Name



verisign.eats.children.thoughtcrime.org

# PKCS #10 SUBJECT

Common Name



iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii. thoughtcrime.org



# PKCS #10 SUBJECT

Common Name



[www.paypal.com\0.thoughtcrime.org](http://www.paypal.com\0.thoughtcrime.org)

# PKCS #10 CERTIFICATE SIGNING REQUEST

CertificateRequest

Version

Subject

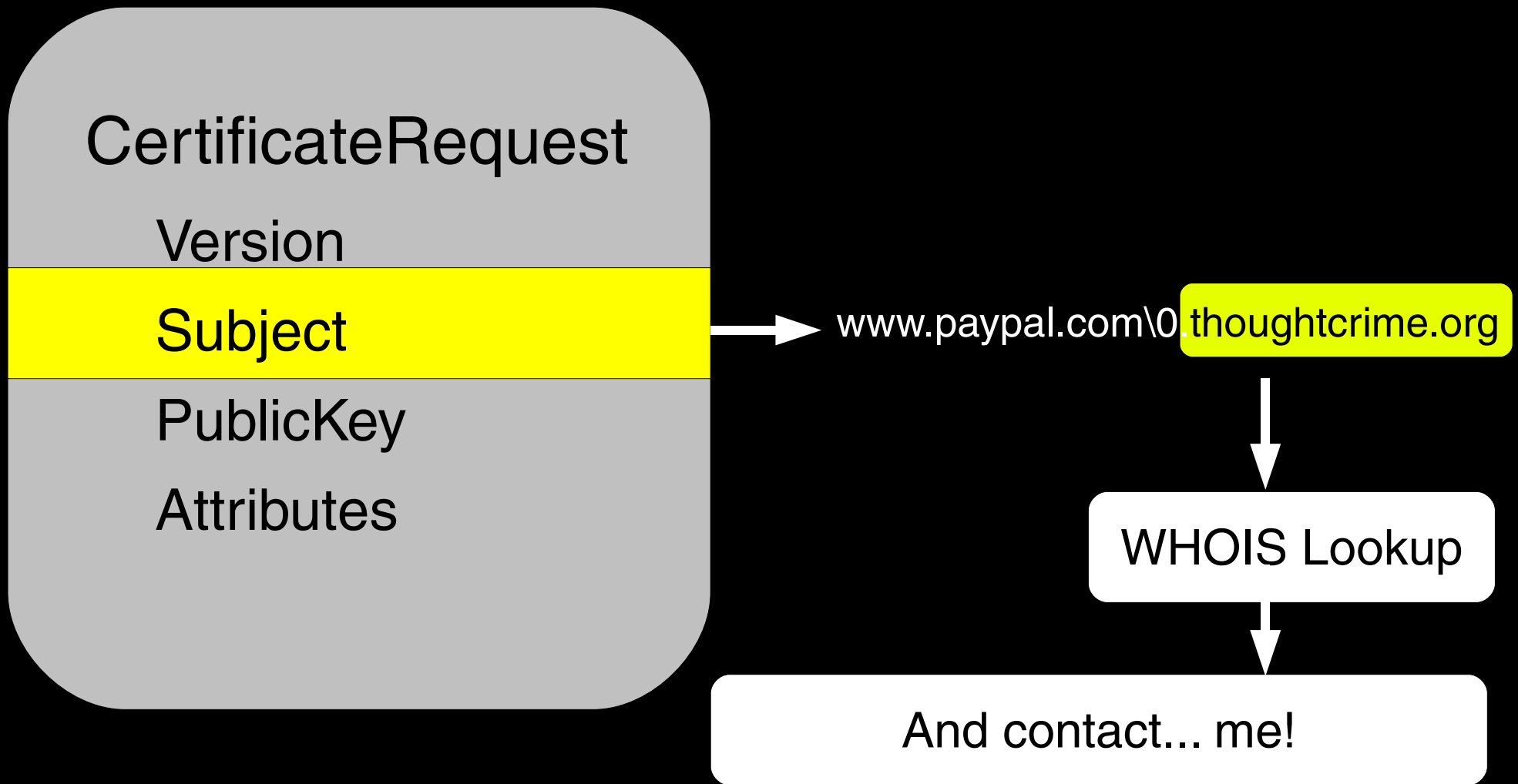
PublicKey

Attributes

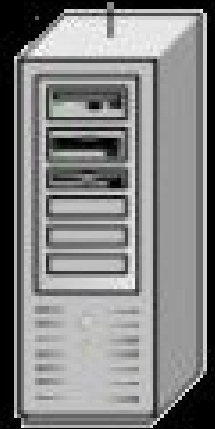


[www.paypal.com\0.thoughtcrime.org](http://www.paypal.com\0.thoughtcrime.org)

# PKCS #10 CERTIFICATE SIGNING REQUEST



# OUR ORIGINAL SCENARIO



Attacker

ClientHello

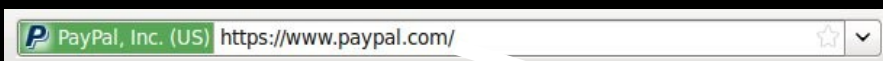


ServerHello,  
ServerCertificate

[www.paypal.com\0.thoughtcrime.org]



# OUR ORIGINAL SCENARIO



X509Certificate

Version

Serial Number

Issuer

Validity

Subject

PublicKey

SignatureAlgorithm

Signature

# OUR ORIGINAL SCENARIO

```
char *destination = getDomainWeAreConnectingTo();  
char *commonName = getCommonNameFromCertificate();  
bool everythingIsOk = (strcmp(destination, commonName) == 0);
```

# IN MEMORY, THOUGH...

char \*destination

w w w . p a y p a l . c o m \0

w w w . p a y p a l . c o m \0 . t h o u g h t c r i m e . o r g \0

char \*commonName

# IN MEMORY, THOUGH...

char \*destination

w w w . p a y p a l . c o m \0

w w w . p a y p a l . c o m \0 . t h o u g h t c r i m e . o r g \0

char \*commonName

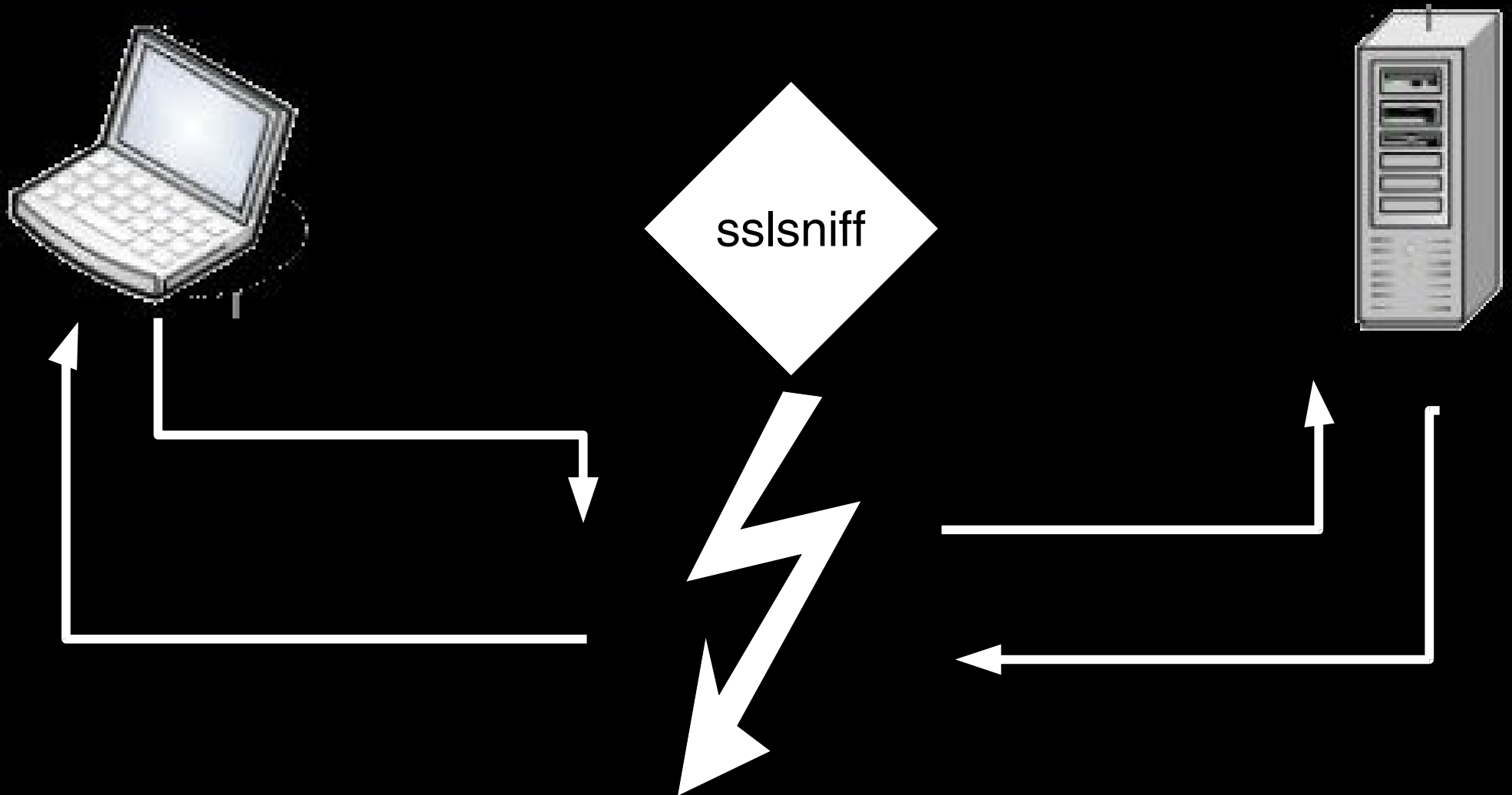


IN THE EYES OF MOST SSL  
IMPLEMENTATIONS, THIS CERTIFICATE IS  
*completely valid* FOR WWW.PAYPAL.COM

# WHAT ARE “MOST” SSL IMPLEMENTATIONS?

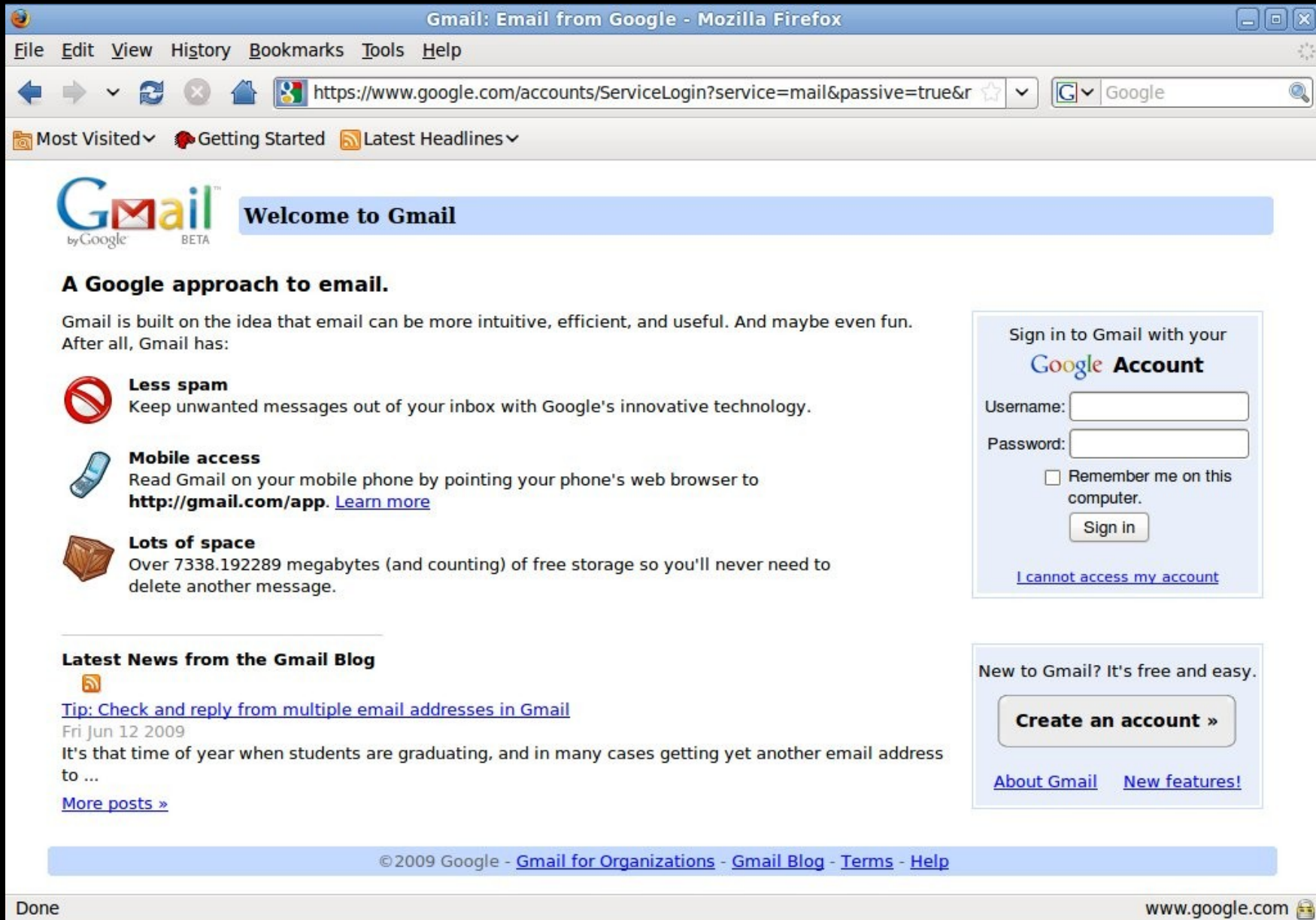
- Web Browsers
  - Firefox (all versions), IE (all versions), Lynx, Curl,
- Mail Clients
  - Thunderbird, Outlook, Evolution
- Chat Clients
  - Pidgin, AIM, irssi, centericq
- SSL VPNs
  - AEP, Citrix, etc...

# A FIRST CUT: UPDATED SSLSNIFF

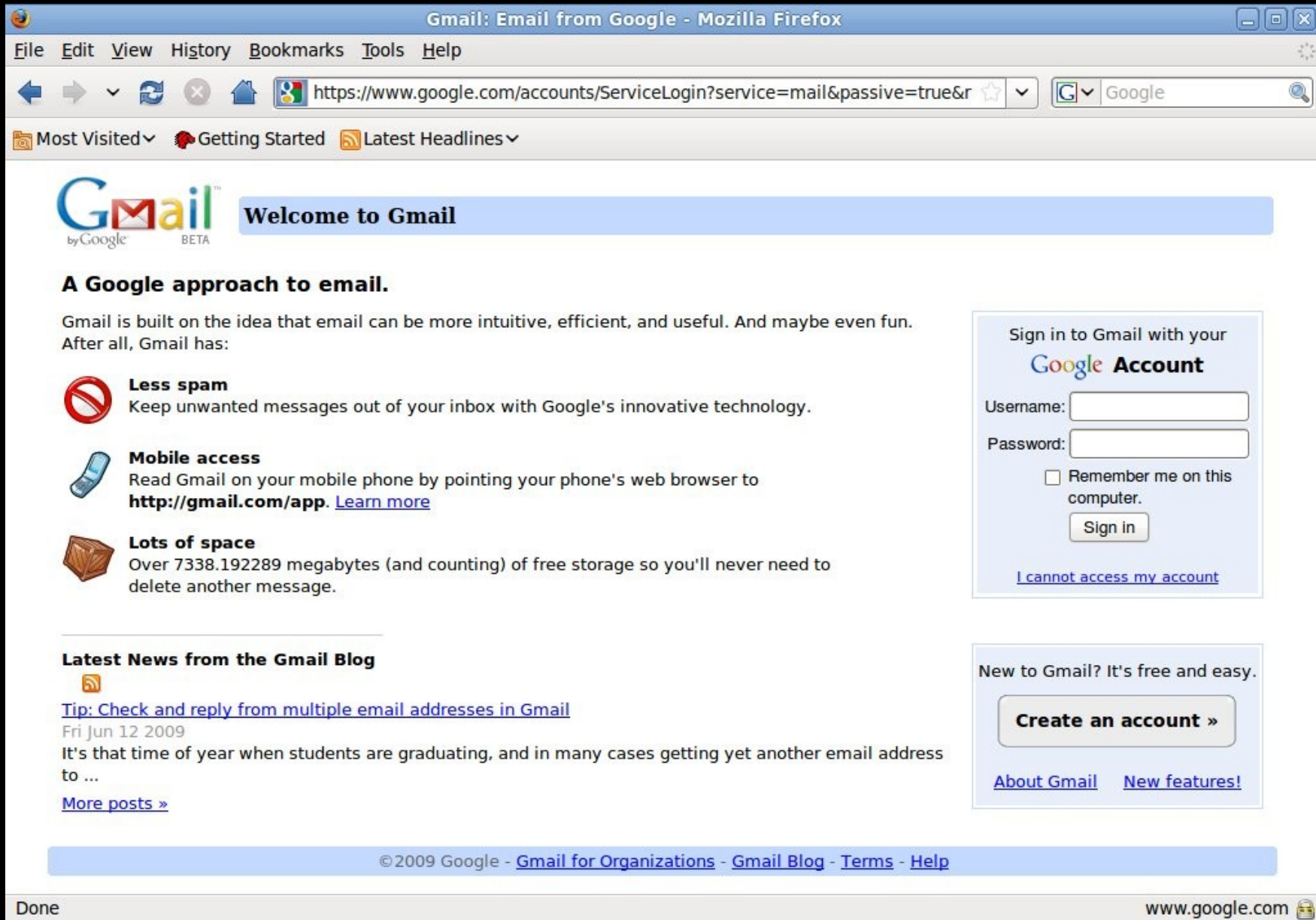


Iff “null prefix attack” certificate  
is available

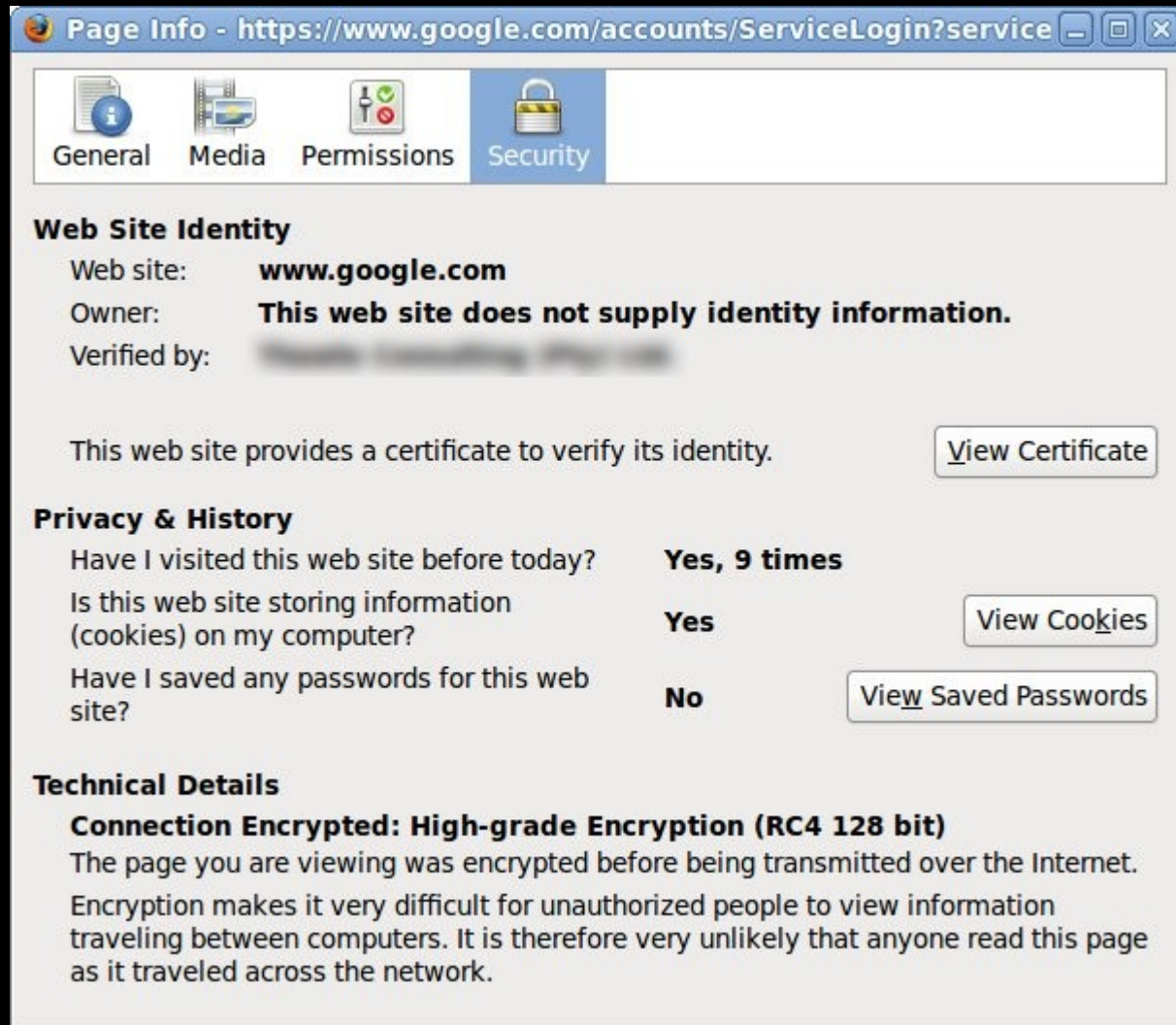
# HOW DOES IT LOOK?



# HOW DOES IT LOOK?

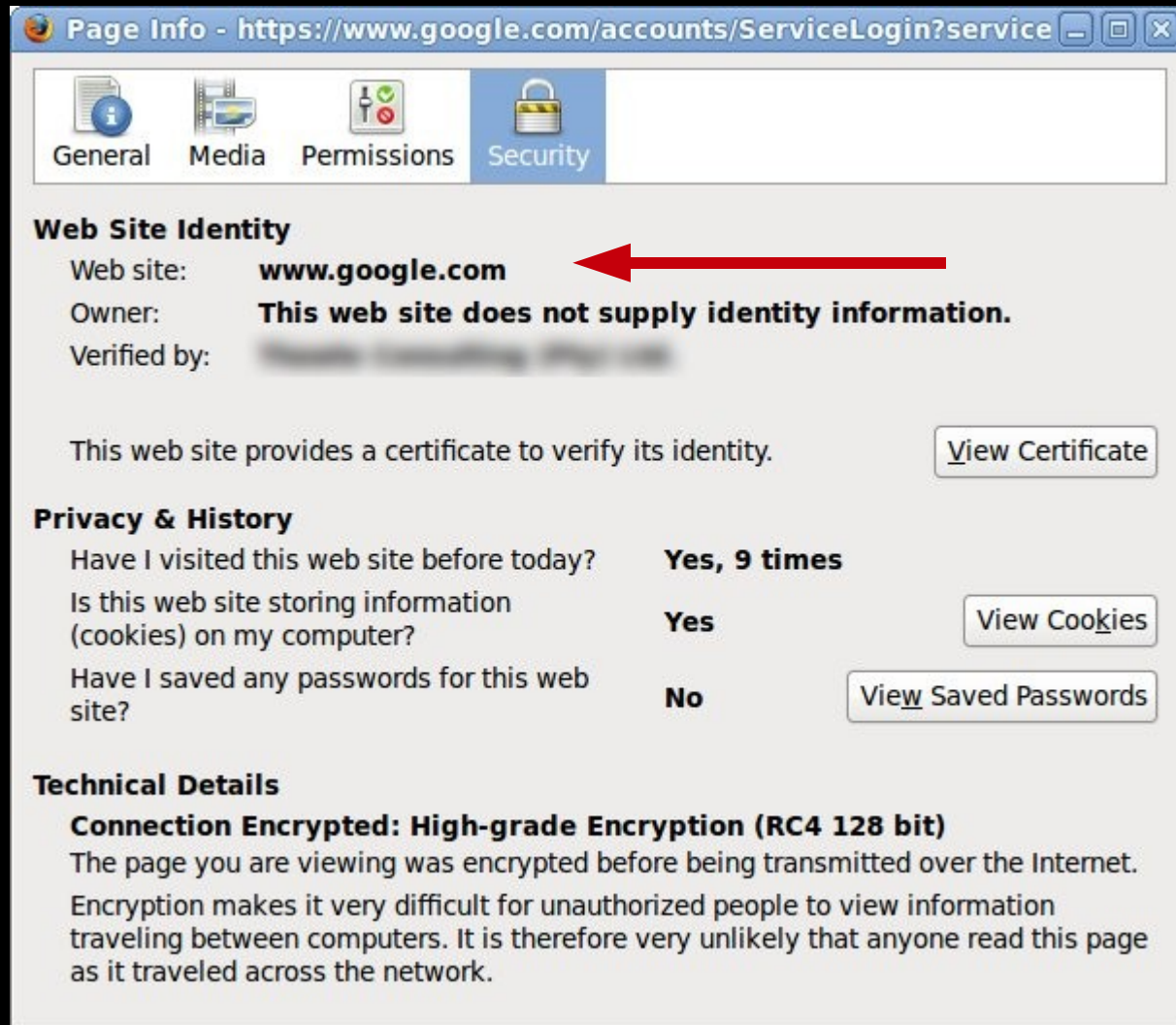


# HOW DOES IT LOOK?





# HOW DOES IT LOOK?



# DISADVANTAGES



I) TARGETED ATTACKS ARE KIND OF LAME.

MAYBE THERE'S ANOTHER TRICK IN HERE  
SOMEWHERE...



```
static int
shexp_match(const char *str, const char *exp, PRBool case_insensitive)
{
    register int x,y;
    int ret,neg;

    ret = 0;
    for(x=0,y=0;exp[y];++y,++x) {
        if(!str[x] && (exp[y] != '(') && (exp[y] != '$') && (exp[y] != '*'))
            ret = ABORTED;
        else {
            switch(exp[y]) {
                case '$':
                    if(str[x])
                        ret = NOMATCH;
                    else
                        --x;          /* we don't want loop to increment x */
                    break;
                case '*':
                    while(exp[++y] == '*') {}
                    if(!exp[y])
                        return MATCH;
                    while(str[x]) {
                        switch(_shexp_match(&str[x++],&exp[y], case_insensitive)) {
                            case NOMATCH:
                                continue;
                            case ABORTED:
                                ret = ABORTED;
                                break;
                            default:
                                return MATCH;
                        }
                    }
                    break;
            }
            if((exp[y] == '$') && (exp[y+1] == '\0') && (!str[x]))
                return MATCH;
            else
                ret = ABORTED;
            break;
        }
        case '[':
            neg = ((exp[++y] == '^') && (exp[y+1] != ']'));
            if(neg)

```

-u:\*\*\* \*scratch\* 6% L46 (C/1 Abbrev)-----

10/22 Copyright © 2014, 2015 Pearson Education, Inc. All rights reserved.



```
static int
_shexp_match(const char *str, const char *exp, PRBool case_insensitive)
{
    register int x,y;
    int ret,neg;

    ret = 0;
    for(x=0,y=0;exp[y];++y,++x) {
        if(!str[x] && (exp[y] != '(') && (exp[y] != '$') && (exp[y] != '*'))
            ret = ABORTED;
        else {
            switch(exp[y]) {
                case '$':
                    if(str[x])
                        ret = NOMATCH;
                    else
                        --x;          /* we don't want loop to increment x */
                    break;
                case '*':
                    while(exp[++y] == '*') {}
                    if(!exp[y])
                        return MATCH;
                    while(str[x]) {
                        switch(_shexp_match(&str[x++],&exp[y], case_insensitive)) {
                            case NOMATCH:
                                continue;
                            case ABORTED:
                                ret = ABORTED;
                                break;
                            default:
                                return MATCH;
                        }
                    }
                    break;
            }
            if((exp[y] == '$') && (exp[y+1] == '\0') && (!str[x]))
                return MATCH;
            else
                ret = ABORTED;
            break;
        }
        case '[':
            neg = ((exp[++y] == '^') && (exp[y+1] != ']'));
            if(neg)

```

-u:\*\*\* \*scratch\* 6% L46 (C/1 Abbrev)-----



```
static int
_shexp_match(const char *str, const char *exp, PRBool case_insensitive)
{
    register int x,y;
    int ret,neg;

    ret = 0;
    for(x=0,y=0;exp[y];++y,++x) {
        if(!str[x] && (exp[y] != '(') && (exp[y] != '$') && (exp[y] != '*'))
            ret = ABORTED;
        else {
            switch(exp[y]) {
                case '$':
                    if(str[x])
                        ret = NOMATCH;
                    else
                        --x; /* we don't want loop to increment x */
                    break;
                case '*':
                    while(exp[++y] == '*') {}
                    if(!exp[y])
                        return MATCH;
                    while(str[x]) {
                        switch(_shexp_match(&str[x++],&exp[y], case_insensitive)) {
                            case NOMATCH:
                                continue;
                            case ABORTED:
                                ret = ABORTED;
                                break;
                            default:
                                return MATCH;
                        }
                    }
                    break;
            }
            if((exp[y] == '$') && (exp[y+1] == '\0') && (!str[x]))
                return MATCH;
            else
                ret = ABORTED;
            break;
        }
        case '[':
            neg = ((exp[++y] == '^') && (exp[y+1] != ']'));
            if (neg)
```

-u:\*\*\* \*scratch\* 6% L46 (C/1 Abbrev)-----



```
static int
_shexp_match(const char *str, const char *exp, PRBool case_insensitive)
{
    register int x,y;
    int ret,neg;

    ret = 0;
    for(x=0,y=0;exp[y];++y,++x) {
        if(!str[x] && (exp[y] != '(') && (exp[y] != '$') && (exp[y] != '*'))
            ret = ABORTED;
        else {
            switch(exp[y]) {
                case '$':
                    if(str[x])
                        ret = NOMATCH;
                    else
                        --x; /* we don't want loop to increment x */
                    break;
                case '*':
                    while(exp[++y] == '*') {}
                    if(!exp[y])
                        return MATCH;
                    while(str[x]) {
                        switch(_shexp_match(&str[x++],&exp[y], case_insensitive)) {
                            case NOMATCH:
                                continue;
                            case ABORTED:
                                ret = ABORTED;
                                break;
                            default:
                                return MATCH;
                        }
                    }
                    break;
            }
            if((exp[y] == '$') && (exp[y+1] == '\0') && (!str[x]))
                return MATCH;
            else
                ret = ABORTED;
            break;
        }
        case '[':
            neg = ((exp[++y] == '^') && (exp[y+1] != ']'));
            if(neg)
```

# UNIVERSAL WILDCARD

\*\0.thoughtcrime.org



# UNIVERSAL WILDCARD

\*~.thoughtcrime.org

# OTHER WEIRD STUFF

- ([www.paypal.com](http://www.paypal.com)|[mail.google.com](http://mail.google.com)|  
[www.etrade.com](http://www.etrade.com)|[www.bankofamerica.com](http://www.bankofamerica.com)|  
[www.wachovia.com](http://www.wachovia.com)|[www.pnc.com](http://www.pnc.com)|  
[www.wellsfargo.com](http://www.wellsfargo.com))\0.thoughtcrime.org

# AND... YOUR REMOTE EXPLOIT.

```
. 144 char *e2 = (char *) PORT_Alloc(sizeof(char)*strlen(exp));
. 145 register int t,p2,p1 = 1;
. 146 int cp;
. 147
. 148 while(1) {
. 149     for(cp=1;exp[cp] != ');cp++)
. 150         if(exp[cp] == '\\')
. 151             ++cp;
. 152     for(p2 = 0;(exp[p1] != '|') && (p1 != cp);p1++,p2++) {
. 153         if(exp[p1] == '\\')
. 154             e2[p2++] = exp[p1++];
. 155         e2[p2] = exp[p1];
. 156     }
. 157     for (t=cp+1; ((e2[p2] = exp[t]) != 0); ++t,++p2) {}
. 158     if(_shexp_match(str,e2, case_insensitive) == MATCH) {
. 159         PORT_Free(e2);
. 160         return MATCH;
. 161     }
. 162     ...
```

# AND... YOUR REMOTE EXPLOIT.

```
. 144 char *e2 = (char *) PORT_Alloc(sizeof(char)*strlen(exp));
. 145 register int t,p2,p1 = 1;
. 146 int cp;
. 147
. 148 while(1) {
. 149     for(cp=1;exp[cp] != ');cp++)
. 150         if(exp[cp] == '\\')
. 151             ++cp;
. 152     for(p2 = 0;(exp[p1] != '|') && (p1 != cp);p1++,p2++) {
. 153         if(exp[p1] == '\\')
. 154             e2[p2++] = exp[p1++];
. 155         e2[p2] = exp[p1];
. 156     }
. 157     for (t=cp+1; ((e2[p2] = exp[t]) != 0); ++t,++p2) {}
. 158     if(_shexp_match(str,e2, case_insensitive) == MATCH) {
. 159         PORT_Free(e2);
. 160         return MATCH;
. 161     }
. 162     ...
```

# AND... YOUR REMOTE EXPLOIT.

```
. 144 char *e2 = (char *) PORT_Alloc(sizeof(char)*strlen(exp));
. 145 register int t,p2,p1 = 1;
. 146 int cp;
. 147
. 148 while(1) {
. 149     for(cp=1;exp[cp] != '\0';cp++)
. 150         if(exp[cp] == '\\')
. 151             ++cp;
. 152     for(p2 = 0;(exp[p1] != '\0') && (p1 != cp);p1++,p2++) {
. 153         if(exp[p1] == '\\')
. 154             e2[p2++] = exp[p1++];
. 155         e2[p2] = exp[p1];
. 156     }
. 157     for (t=cp+1; ((e2[p2] = exp[t]) != 0); ++t,++p2) {}
. 158     if(_shexp_match(str,e2, case_insensitive) == MATCH) {
. 159         PORT_Free(e2);
. 160         return MATCH;
. 161     }
. 162     ...
```

# AND... YOUR REMOTE EXPLOIT.

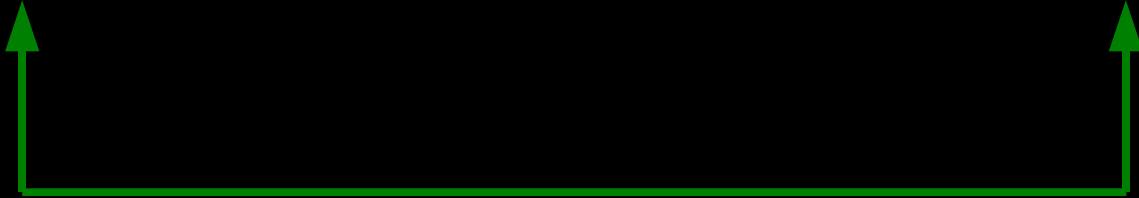
```
. 144 char *e2 = (char *) PORT_Alloc(sizeof(char)*strlen(exp));
. 145 register int t,p2,p1 = 1;
. 146 int cp;
. 147
. 148 while(1) {
. 149     for(cp=1;exp[cp] != '\0';cp++)
. 150         if(exp[cp] == '\\')
. 151             ++cp;
. 152     for(p2 = 0;(exp[p1] != '\0') && (p1 != cp);p1++,p2++) {
. 153         if(exp[p1] == '\\')
. 154             e2[p2++] = exp[p1++];
. 155         e2[p2] = exp[p1];
. 156     }
. 157     for (t=cp+1; ((e2[p2] = exp[t]) != 0); ++t,++p2) {}
. 158     if(_shexp_match(str,e2, case_insensitive) == MATCH) {
. 159         PORT_Free(e2);
. 160         return MATCH;
. 161     }
. 162     ...
```

AND... YOUR REMOTE EXPLOIT.

(AAAAAAAAAAAAAAAAAAAAAAAAAAAA\0OVERWRITE).foo.com

# AND... YOUR REMOTE EXPLOIT.

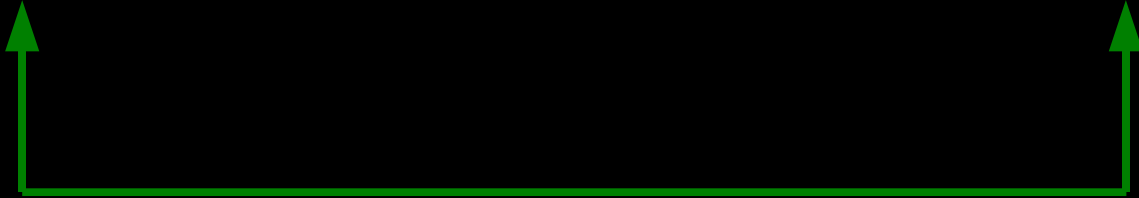
(AAAAAAAAAAAAAAAAAAAAAAAAAAAA\0OVERWRITE).foo.com





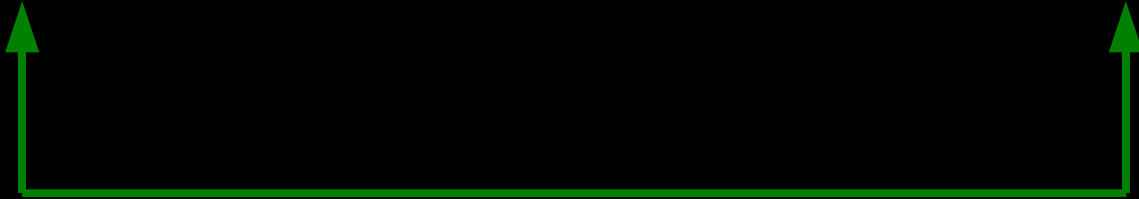
# AND... YOUR REMOTE EXPLOIT.

(AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\0OVERWRITE).foo.com



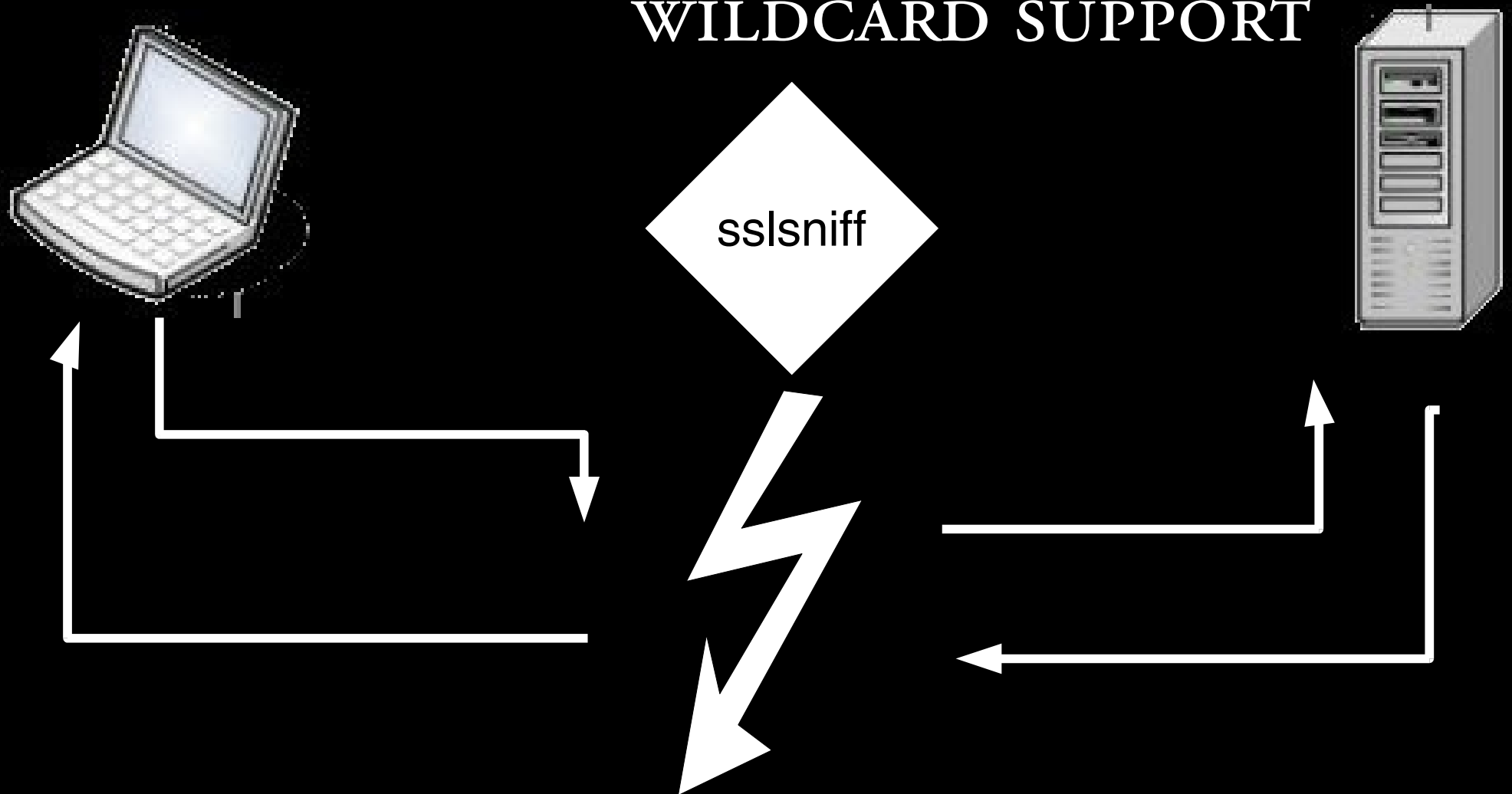
# AND... YOUR REMOTE EXPLOIT.

(AAAAAAAAAAAAAAAAAAAAAAAAAAAA\0OVERWRITE).foo.com



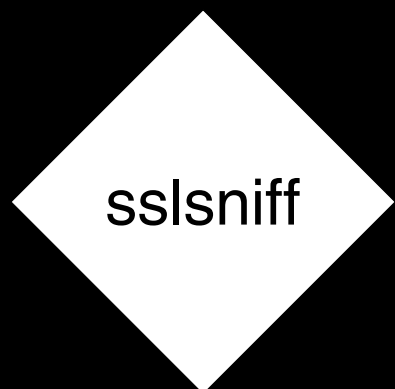
- ✓ No signed signature required!
- ✓ Possible to sneak non-ASCII characters past the NSS filters.
- ✓ This yields something exploitable in Firefox, Thunderbird, Evolution, Pidgin, and AIM.

# A SECOND CUT: SSLSNIFF WITH WILDCARD SUPPORT



- Perform MITM if “null termination attack” cert is available.
- Or perform MITM with “universal wildcard” cert if client is NSS.

# A SECOND CUT: UPDATED SSLSNIFF



- Watches network and fingerprints clients for level of vulnerability.
- Every NSS client's communication is intercepted – either with a specific “null termination” certificate, or with the “universal wildcard” certificate.
- Every non-NSS client that is vulnerable is intercepted with a “null termination” certificate if available for the destination host.
- Non-vulnerable clients are left alone to avoid detection.

WHAT DO WE HAVE TO WORRY ABOUT?

# WHAT DO WE HAVE TO WORRY ABOUT?

1) Certificate Revocation

2) Updates

# WHAT DO WE HAVE TO WORRY ABOUT?

## 1) Certificate Revocation

- It would be unfortunate if some bitter Certificate Authority decided to revoke our universal wildcard certificates or any of our null-termination certificates.

## 2) Updates

- It would be unfortunate if some bitter SSL implementation decided to start paying attention to how ASN.1 is formatted.

# WHAT DO WE HAVE TO WORRY ABOUT?

## 1) Certificate Revocation

- These days, it's all about Online Certificate Status Protocol (OCSP).
- Whenever a SSL stack sees a new certificate, it makes a quick request to the OCSP URL that the signing CA embedded in it.
- The SSL stack receives a signed response from the OCSP provider indicating whether the certificate has been revoked or not.



Certificate:

Data:

Version: 3 (0x2)

Serial Number:

01:2a:39:76:0d:3f:4f:c9:0b:e7:bd:2b:cf:95:2e:7a

Signature Algorithm: sha1WithRSAEncryption

Issuer: C=ZA, O=Thawte Consulting (Pty) Ltd., CN=Thawte SGC CA

Validity

Not Before: Mar 27 22:20:07 2009 GMT

Not After : Mar 27 22:20:07 2010 GMT

Subject: C=US, ST=California, L=Mountain View, O=Google Inc, CN=www.google.com

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

RSA Public Key: (1024 bit)

Modulus (1024 bit):

00:d6:b9:e1:ad:b8:61:0b:1f:4e:b6:3c:09:3d:ab:  
e8:e3:2b:b6:e8:a4:3a:78:2f:d3:51:20:22:45:95:  
d8:00:91:33:9a:a7:a2:48:ea:30:57:26:97:66:c7:  
5a:ef:f1:9b:0c:3f:e1:b9:7f:7b:c3:c7:cc:af:9c:  
d0:1f:3c:81:15:10:58:fc:06:b3:bf:bc:9c:02:b9:  
51:dc:fb:a6:b9:17:42:e6:46:e7:22:cf:6c:27:10:  
fe:54:e6:92:6c:0c:60:76:9a:ce:f8:7f:ac:b8:5a:  
08:4a:dc:b1:64:bd:a0:74:41:b2:ac:8f:86:9d:1a:  
de:58:09:fd:6c:0a:25:e0:79

Exponent: 65537 (0x10001)

X509v3 extensions:

X509v3 Extended Key Usage:

TLS Web Server Authentication, TLS Web Client Authentication, Netscape Server Gated Crypto

X509v3 CRL Distribution Points:

URI:http://crl.thawte.com/ThawteSGCCA.crl

Authority Information Access:

OCSP - URI:http://ocsp.thawte.com

CA Issuers - URI:http://www.thawte.com/repository/Thawte\_SGC\_CA.crt

X509v3 Basic Constraints: critical

CA:FALSE

Signature Algorithm: sha1WithRSAEncryption

39:b6:fb:11:bc:33:2c:c3:90:48:e3:6e:c3:9b:38:b1:42:d1:  
00:09:58:63:a0:e1:98:1c:85:f2:ef:10:1d:60:4e:51:09:62:  
f5:05:bd:9d:4f:87:6c:98:72:07:80:c3:59:48:14:e2:d6:ef:  
d0:8f:33:6a:68:31:fa:b7:bb:85:cc:f7:c7:47:7b:67:93:3c:  
c3:16:51:9b:6f:87:20:fd:67:4c:2b:ea:6a:49:db:11:d1:bd:  
d7:95:22:43:7a:06:7b:4e:f6:37:8e:a2:b9:cf:1f:a5:d2:bd:  
3b:04:97:39:b3:0f:fa:38:b5:af:55:20:88:60:93:f2:de:db:

# DEFEATING OCSP

```
OCSPResponse ::= SEQUENCE {  
    responseStatus      OCSPResponseStatus,  
    responseBytes       [0] EXPLICIT ResponseBytes OPTIONAL  
}
```

# DEFEATING OCSP

```
OCSPResponse ::= SEQUENCE {  
    responseStatus      OCSPResponseStatus,  
    responseBytes       [0] EXPLICIT ResponseBytes OPTIONAL  
}
```

```
ResponseBytes ::= SEQUENCE {  
    responseType OBJECT IDENTIFIER,  
    response      OCTET STRING  
}
```

```
BasicOCSPResponse ::= SEQUENCE {  
    tbsResponseData ResponseData,  
    signatureAlgorithm AlgorithmIdentifier,  
    signature        BIT STRING,  
    certs            [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL }
```

# DEFEATING OCSP

```
OCSPResponse ::= SEQUENCE {  
    responseStatus      OCSPResponseStatus,  
    responseBytes       [0] EXPLICIT ResponseBytes OPTIONAL  
}
```

```
ResponseBytes ::= SEQUENCE {  
    responseType OBJECT IDENTIFIER,  
    response      OCTET STRING  
}
```

```
BasicOCSPResponse ::= SEQUENCE {  
    tbsResponseData  ResponseData,  
    signatureAlgorithm AlgorithmIdentifier,  
    signature        BIT STRING,  
    certs            [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL }
```

# DEFEATING OCSP

```
OCSPResponse ::= SEQUENCE {  
    responseStatus      OCSPResponseStatus,  
    responseBytes       [0] EXPLICIT ResponseBytes OPTIONAL  
}
```

```
ResponseBytes ::= SEQUENCE {  
    responseType OBJECT IDENTIFIER,  
    response      OCTET STRING  
}
```

```
BasicOCSPResponse ::= SEQUENCE {  
    tbsResponseData  ResponseData,  
    signatureAlgorithm AlgorithmIdentifier,  
    signature         BIT STRING,  
    certs             [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL }
```

# DEFEATING OCSP

```
OCSPResponse ::= SEQUENCE {  
    responseStatus      OCSPResponseStatus,  
    responseBytes       [0] EXPLICIT ResponseBytes OPTIONAL  
}
```

```
ResponseBytes ::= SEQUENCE {  
    responseType OBJECT IDENTIFIER,  
    response      OCTET STRING  
}
```

```
BasicOCSPResponse ::= SEQUENCE {  
    tbsResponseData ResponseData,  
    signatureAlgorithm AlgorithmIdentifier,  
    signature        BIT STRING,  
    certs             [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL }
```

# DEFEATING OCSP

```
OCSPResponse ::= SEQUENCE {
```

```
    responseStatus      OCSPResponseStatus,
```

```
    responseBytes       [0] EXPLICIT ResponseBytes OPTIONAL
```

```
}
```

```
ResponseBytes ::= SEQUENCE {
```

```
    responseType OBJECT IDENTIFIER,
```

```
    response      OCTET STRING
```

```
}
```

```
BasicOCSPResponse ::= SEQUENCE {
```

```
    tbsResponseData ResponseData,
```

```
    signatureAlgorithm AlgorithmIdentifier,
```

```
    signature      BIT STRING,
```

```
    certs          [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL }
```



# DEFEATING OCSP

OCSPResponse ::= SEQUENCE {

    responseStatus      OCSPResponseStatus,

    responseBytes      [0] EXPLICIT ResponseBytes OPTIONAL

}

OCSPResponseStatus ::= ENUMERATED {

    successful          (0), --Response has valid confirmations

    malformedRequest    (1), --Illegal confirmation request

    internalError        (2), --Internal error in issuer

    tryLater            (3), --Try again later

                        --(4) is not used

    sigRequired          (5), --Must sign the request

    unauthorized        (6) --Request unauthorized

}





# DEFEATING OCSP

OCSPResponse ::= SEQUENCE {

    responseStatus      OCSPResponseStatus,

    responseBytes      [0] EXPLICIT ResponseBytes OPTIONAL

}

OCSPResponseStatus ::= ENUMERATED {

    successful          (0), --Response has valid confirmations

    malformedRequest    (1), --Illegal confirmation request

    internalError        (2), --Internal error in issuer

    tryLater            (3), --Try again later

                        --(4) is not used

    sigRequired          (5), --Must sign the request

    unauthorized        (6) --Request unauthorized

}



# DEFEATING OCSP

```
OCSPResponse ::= SEQUENCE {  
    responseStatus      OCSPResponseStatus = 3,  
    responseBytes       [0] EXPLICIT ResponseBytes OPTIONAL  
}
```

# DEFEATING OCSP

```
OCSPResponse ::= SEQUENCE {  
    responseStatus      OCSPResponseStatus = 3,  
  
}
```

# PROPOSED STANDARD

Network Working Group  
Request for Comments: 2560  
Category: Standards Track

M. Myers  
VeriSign  
R. Ankney

CertCo  
A. Malpani  
ValiCert

June 1999

X.509 Internet Public Key Infrastructure

Online Certificate Status Protocol - OCSP

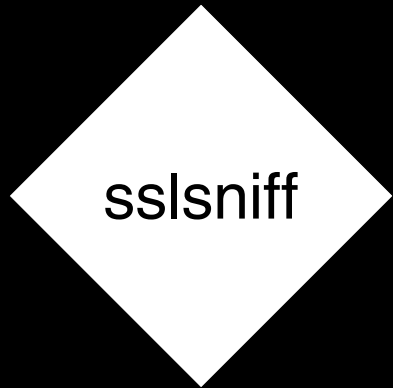
Status of this Memo

This document specifies an Internet Standards Track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (1999). All Rights Reserved.

# A THIRD CUT: OCSP-AWARE SSLSNIFF



- Watch network and fingerprints clients for level of vulnerability.
- Every NSS client's communication is intercepted – either with a specific “null termination” certificate, or with the “universal wildcard” certificate.
- Every non-NSS client that is vulnerable is intercepted with a “null termination” certificate if available for the destination host.
- Non-vulnerable clients are left alone to avoid detection.
- Optionally watch for OCSP requests corresponding to certificates we're using, and “tryLater” them to defeat OCSP.

# WHAT DO WE HAVE TO WORRY ABOUT?

## 2) Updates

- It used to be that people, you know, downloaded and installed updates.
- As software gets more complicated, it is inevitably shipped with more bugs, and attackers are situated to exploit them on a larger scale.
- So some have felt the need to deploy self-updating software in order to fix problems rapidly.

# WHAT DO WE HAVE TO WORRY ABOUT?

## 2) Updates

- This is bad news for us, because by standing here and talking to you about this stuff, it probably means that SSL implementations are going to fix these problems.
- But their update mechanisms in themselves seem like kind of a dangerous idea, right?
- Maybe there's something we can do about our problem.

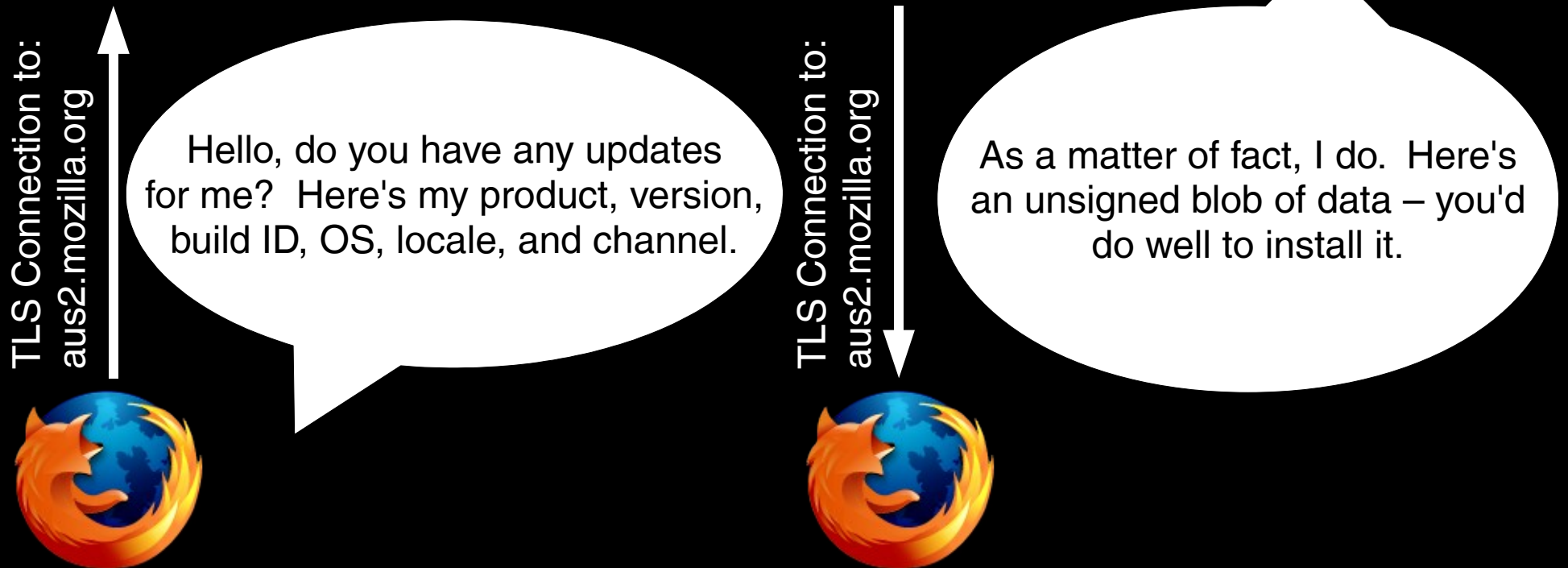
# FIREFOX/THUNDERBIRD: A CASE STUDY

- When you install Firefox, it comes with a feature called “automatic update service,” which happens to be enabled by default.
- Here be dragons.



# FIREFOX/THUNDERBIRD: A CASE STUDY

## Update Server In The Sky



# FIREFOX/THUNDERBIRD: A CASE STUDY

- Firefox and Thunderbird depend on their TLS connection to the update server to defend them against all possible attacks.
- Code is returned from the update server either as a binary diff against the distribution binary the client is running, or as a complete image of the binary.
- By default, “minor” updates are downloaded and installed silently – only prompting the user to restart their browser once everything is done.
  - The update server is the one who reports the version number of the update, so it is effectively up to the server whether the image it provides is installed silently or not.

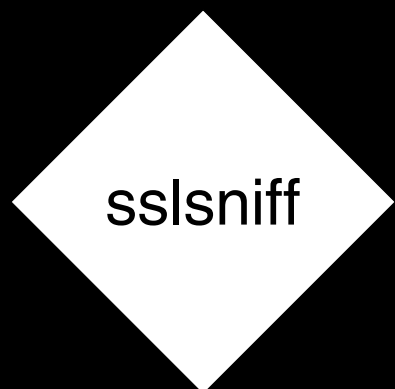
# FIREFOX/THUNDERBIRD: A CASE STUDY

- As vendors start to release patches for this vulnerability, the update mechanisms themselves will be vulnerable.
- All we need is a universal wildcard cert, or alternately a null-termination prefix cert for aus2.mozilla.org, and we can take control of the update mechanism to deliver payloads of our choice.
  - This could be anything:
    - A rootkit that logs keystrokes.
    - Something that sends all traffic/email through a server of our choosing.
    - A completely legitimate image that just happens to include our own CA certs.
    - Or, just to be confusing, a totally different web browser (“Thank you for updating to Galeon 0.0.3!”) or even a completely different type of application – notepad.exe comes to mind.

# FIREFOX/THUNDERBIRD: A CASE STUDY

- In order to patch your system effectively, you will not be able to trust anything that comes through automatic updates.

# A FOURTH CUT: UPDATE-AWARE SSLSNIFF



- Watch network and fingerprints clients for level of vulnerability.
- Every NSS client's communication is intercepted – either with a specific “null prefix” certificate, or with the “universal wildcard” certificate.
- Every non-NSS client that is vulnerable is intercepted with a “null prefix” certificate if available for the destination host.
- Non-vulnerable clients are left alone to avoid detection.
- Optionally watch for OCSP requests corresponding to certificates we're using, and “tryLater” them to defeat OCSP.
- Optionally watch for Firefox/Thunderbird update polls, and respond with a “custom” build.

# POSTSCRIPT:

## STRIPPING NULL IS NO SOLUTION

- Some SSL/TLS implementations (Safari, Opera) appear to strip '\0' from commonName strings before comparing.

- Thus:

`www.paypal.com\0.thoughtcrime.org`

- Becomes:

`www.paypal.com.thoughtcrime.org`

# POSTSCRIPT:

## STRIPPING NULL IS NO SOLUTION

- These implementations are vulnerable to a variation of our attack.
- The key is that some Certificate Authorities are vulnerable to this attack internally.
  - When presented with `www.paypal.com\0.thoughtcrime.org`, some CAs internally validate it as `www.paypal.com`
  - But the whole string (`www.paypal.com\0.thoughtcrime.org`) is what ends up in the subject of the cert they later issue.

# POSTSCRIPT:

## STRIPPING NULL IS NO SOLUTION

- So if we register a domain like sitekey.ba
- We can get a certificate for sitekey.ba\0nkofamerica.com
- The CAs that are internally vulnerable to this attack will validate that certificate against sitekey.ba, which we own.
- When the cert is later presented to a SSL implementation that strips \0, the certificate's common name becomes: sitekey.bankofamerica.com



# CONCLUSION

- We have a MITM attack that will intercept communication for almost all SSL/TLS implementations.
- In the case of NSS (Firefox, Thunderbird, Evolution, AIM, Pidgin) we only need a single certificate.
- We've defeated the OCSP protocol as implemented.
- We've hijacked the Mozilla auto-updates for both applications and extensions.
- We've got an exploitable overflow.
- In short, we've got your passwords, your communication, and control over your computer.