CSE 484 (Winter 2008)

Applied Cryptography

Tadayoshi Kohno

Thanks to Dan Boneh, Dieter Gollmann, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Goals for Today

- Asymptric cryptography
- Project 2 out
  - Informal checkpoint: Feb 22 (11:59pm)
  - Full submission: Feb 29 (11:59pm)

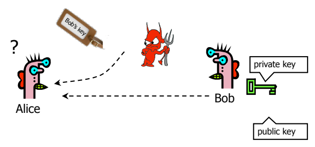
Advantages of Public-Key Crypto

- Confidentiality without shared secrets
  - Very useful in open environments
  - No "chicken-and-egg" key establishment problem
    - With symmetric crypto, two parties must share a secret before they can exchange secret messages
    - Caveats to come
- Authentication without shared secrets
  - Use digital signatures to prove the origin of messages
- Reduce protection of information to protection of authenticity of public keys
  - No need to keep public keys secret, but must be sure that Alice's public key is <u>really</u> her true public key

Disadvantages of Public-Key Crypto

- Calculations are 2-3 orders of magnitude slower
  - Modular exponentiation is an expensive computation
  - Typical usage: use public-key cryptography to establish a shared secret, then switch to symmetric crypto
    - We'll see this in IPSec and SSL
- Keys are longer
  - 1024 bits (RSA) rather than 128 bits (AES)
- Relies on unproven number-theoretic assumptions
  - What if factoring is easy?
    - Factoring is <u>believed</u> to be neither P, nor NP-complete
  - (Of course, symmetric crypto also rests on unproven assumptions)

Authenticity of Public Keys



Problem: How does Alice know that the public key she received is really Bob's public key?

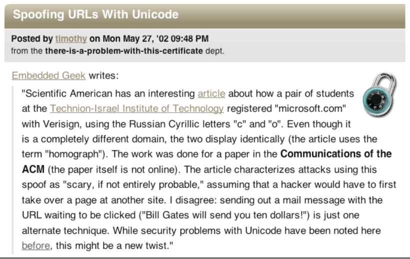
Distribution of Public Keys

- Public announcement or public directory
  - Risks: forgery and tampering
- Public-key certificate
  - Signed statement specifying the key and identity
    - $\text{sig}_{\text{Alice}}$("Bob", PK_B)
- Common approach: certificate authority (CA)
  - Single agency responsible for certifying public keys
  - After generating a private/public key pair, user proves his identity and knowledge of the private key to obtain CA's certificate for the public key (offline)
  - Every computer is pre-configured with CA's public key

Hierarchical Approach

- Single CA certifying every public key is impractical
- Instead, use a trusted root authority
  - For example, Verisign
  - Everybody must know the public key for verifying root authority's signatures
- Root authority signs certificates for lower-level authorities, lower-level authorities sign certificates for individual networks, and so on
  - Instead of a single certificate, use a certificate chain
    - $\text{sig}_{\text{Verisign}}$("UW", PK_{UW}), sig_{UW}("Alice", PK_A)
  - What happens if root authority is ever compromised?

Many Challenges



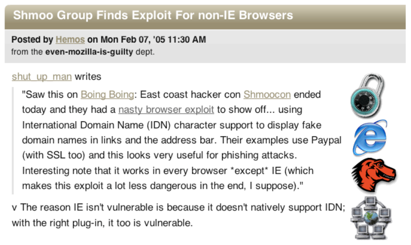
Spoofing URLs With Unicode

Posted by timothy on Mon May 27, '02 09:48 PM
from the **there-is-a-problem-with-this-certificate** dept.

Embedded Geek writes:

"Scientific American has an interesting article about how a pair of students at the Technion-Israel Institute of Technology registered "microsoft.com" with Verisign, using the Russian Cyrillic letters "c" and "o". Even though it is a completely different domain, the two display identically (the article uses the term "homograph"). The work was done for a paper in the **Communications of the ACM** (the paper itself is not online). The article characterizes attacks using this spoof as "scary, if not entirely probable," assuming that a hacker would have to first take over a page at another site. I disagree: sending out a mail message with the URL waiting to be clicked ("Bill Gates will send you ten dollars!") is just one alternate technique. While security problems with Unicode have been noted here before, this might be a new twist."

Many Challenges


Shmoo Group Finds Exploit For non-IE Browsers

Posted by Hemos **on Mon Feb 07, '05 11:30 AM**
from the **even-mozilla-is-guilty** dept.

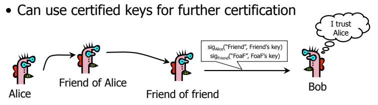
shut_up_man writes

"Saw this on Boing Boing: East coast hacker con Shmoocon ended today and they had a nasty browser exploit to show off... using International Domain Name (IDN) character support to display fake domain names in links and the address bar. Their examples use Paypal (with SSL too) and this looks very useful for phishing attacks. Interesting note that it works in every browser *except* IE (which makes this exploit a lot less dangerous in the end, I suppose)."

v The reason IE isn't vulnerable is because it doesn't natively support IDN; with the right plug-in, it too is vulnerable.

Alternative: "Web of Trust"

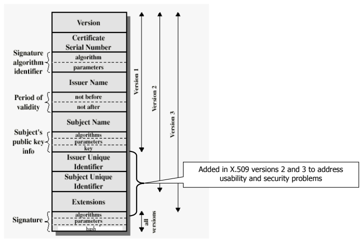
- Used in PGP (Pretty Good Privacy)
- Instead of a single root certificate authority, each person has a set of keys they "trust"
  - If public-key certificate is signed by one of the "trusted" keys, the public key contained in it will be deemed valid
- Trust can be transitive
  - Can use certified keys for further certification


Alice
Friend of Alice
Friend of friend
sig_Alice("Friend", Friend's key)
sig_Friend("FoaF", FoaF's key)
I trust Alice
Bob

X.509 Authentication Service

- Internet standard (1988-2000)
- Specifies certificate format
  - X.509 certificates are used in IPSec and SSL/TLS
- Specifies certificate directory service
  - For retrieving other users' CA-certified public keys
- Specifies a set of authentication protocols
  - For proving identity using public-key signatures
- Does not specify crypto algorithms
  - Can use it with any digital signature scheme and hash function, but hashing is required before signing

X.509 Certificate



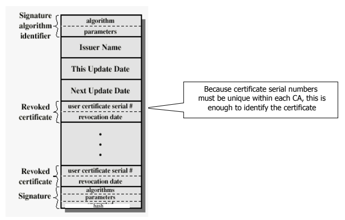
Certificate Revocation

- Revocation is <u>very</u> important
- Many valid reasons to revoke a certificate
  - Private key corresponding to the certified public key has been compromised
  - User stopped paying his certification fee to this CA and CA no longer wishes to certify him
  - CA's certificate has been compromised!
- Expiration is a form of revocation, too
  - Many deployed systems don't bother with revocation
  - Re-issuance of certificates is a big revenue source for certificate authorities

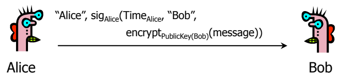
Certificate Revocation Mechanisms

- Online revocation service
  - When a certificate is presented, recipient goes to a special online service to verify whether it is still valid
    - Like a merchant dialing up the credit card processor
- Certificate revocation list (CRL)
  - CA periodically issues a signed list of revoked certificates
    - Credit card companies used to issue thick books of canceled credit card numbers
  - Can issue a "delta CRL" containing only updates
- Question: does revocation protect against forged certificates?

X.509 Certificate Revocation List

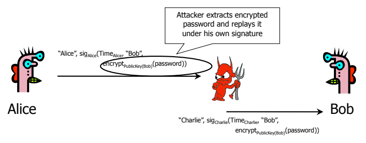

Signature algorithm identifier
algorithm
parameters
Issuer Name
This Update Date
Next Update Date
Revoked certificate
user certificate serial #
revocation date
Revoked certificate
user certificate serial #
revocation date
Signature
algorithms
parameters
hash
Because certificate serial numbers must be unique within each CA, this is enough to identify the certificate

X.509 Version 1


"Alice", sig$_{Alice}$(Time$_{Alice}$, "Bob", encrypt$_{PublicKey(Bob)}$(message))
Alice
Bob

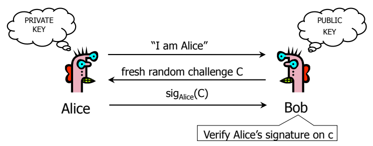
- Encrypt, then sign for authenticated encryption
  - Goal: achieve both confidentiality and authentication
  - E.g., encrypted, signed password for access control
- Does this work?

Attack on X.509 Version 1


Attacker extracts encrypted password and replays it under his own signature
"Alice", sig_{Alice}(Time$_{Alice}$, "Bob", encrypt$_{PublicKey(Bob)}$(password))
Alice
Bob
"Charlie", $sig_{Charlie}$(Time$_{Charlie}$, "Bob", encrypt$_{PublicKey(Bob)}$(password))

◆ Receiving encrypted password under signature does <u>not</u> mean that the sender actually knows the password!

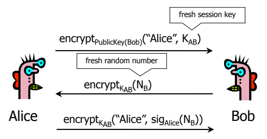
Authentication with Public Keys



1. Only Alice can create a valid signature
2. Signature is on a fresh, unpredictable challenge

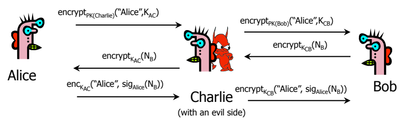
Potential problem: Alice will sign <u>anything</u>

Early Version of SSL (Simplified)


fresh session key
encrypt$_{PublicKey(Bob)}$("Alice", K_{AB})
fresh random number
encrypt$_{KAB}$(N_B)
Alice
Bob
encrypt$_{KAB}$("Alice", sig$_{Alice}$(N_B))

◆ Bob's reasoning: I must be talking to Alice because…
- Whoever signed N_B knows Alice's private key… Only Alice knows her private key… Alice must have signed N_B… N_B is fresh and random and I sent it encrypted under K_{AB}… Alice could have learned N_B only if she knows K_{AB}… She must be the person who sent me K_{AB} in the first message…

Breaking Early SSL


encrypt$_{PK(Charlie)}$("Alice",K_{AC})
encrypt$_{KAC}$(N_B)
enc$_{KAC}$("Alice", sig$_{Alice}$(N_B))
Alice
Charlie
(with an evil side)
encrypt$_{PK(Bob)}$("Alice",K_{CB})
encrypt$_{KCB}$(N_B)
encrypt$_{KCB}$("Alice", sig$_{Alice}$(N_B))
Bob

◆ Charlie uses his legitimate conversation with Alice to impersonate Alice to Bob
- Information signed by Alice is not sufficiently explicit

Programming Project #2

- Out today, Monday, Feb 11
- Due Friday, Feb 29, 11:59pm
  - Submit via Catalyst system
- Teams of up to three people
  - New teams OK (old teams also OK)

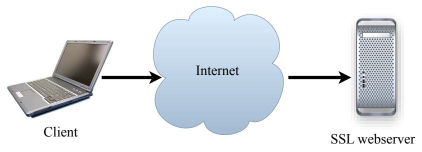
- Basic idea: Implement a "Man-in-the-Middle" attack against SSL
- Recall Security and Privacy Code of Ethics form
- Based on Dan Boneh's CS255 project (Stanford)
  - Slides: http://crypto.stanford.edu/~dabo/cs255/proj2_pres.pdf

Overview

- MITM attack against SSL
  - Not at network layer (not re-writing packets, etc)
  - At SSL Proxy Layer, in Java
    - Networking
    - SSL
    - Certificates
- Password-based authentication for MITM server
  - Hashed, salted passwords
  - Password file encrypted with an authenticated encryption scheme.

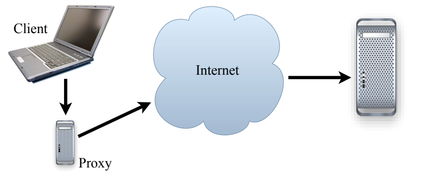
Overview

- Normal SSL
  - SSL encrypted data routed like normal TCP/IP over Internet



Proxy Server

- Browser connects to proxy
- Proxy connects to web server and forwards between the two



“Man in the Middle”

- Instead of forwarding encrypted data between the two hosts, the proxy will set up two different SSL connections
  - Proxy <--> Remote Server
    - Normal SSL client connection to remote site
  - Proxy <--> Browser
    - SSL server connection to the browser, using its own certificate, with some data cloned from the remote hosts’ certificate
    - If browser accepts this fake certificate, the proxy has access to the data in the clear!

What we provided

- Basic Proxy Server setup
  - Parses CONNECT request and sets up a connection between client and remote server
- Basic Admin Server/Client
  - Server listens for connections on plain socket and parses out username/password/command that client sends

Basic Admin Server/Client

- Goal: Experience in adding security features to an application
  - Secure connection between admin client and proxy server using SSL
  - Password based authentication for client
    - Secure storage of password file (authenticated encryption)
    - Passwords stored, hashed, using public and private salt

Proxy Server

- Already listens for browser CONNECT requests and sets up the needed SSL connections
- You should
  - Understand the connections being made
  - Obtain the remote server certificate from the remote SSL connection
  - Copy the relevant fields and sign a forged certificate using your CA cert (from your keystore); use IAIK
  - Modify the code creating the client SSL connection to use the newly forged certificate

Signing Certificate

- Build a self-signed certificate for the proxy server (the proxy server's "CA" certificate)
  - keytool -genkey -keyalg RSA
  - Store this in a JKS keystore for use by your proxy server
  - Use it for signing your programmatically generated certs
  - Your proxy pretends to be the CA
- Submit a keystore with your project

Generating Certs "On the Fly"

- Not easy to generate certificates programmatically using standard Java libraries
- Instead, use the IAIK-JCE library
  - iaik.x509.X509Certificate (class)

iaik.x509.X509Certificate

- To convert from a java certificate:
  - new X509Certificate(javaCert.getEncoded());
- Signing
  - cert.sign(AlgorithmID.sha256withRSAEncryption, issuerPk);
- See iaik.asn1.structures.Name
  - For extracting info (e.g., common name) from the certificate's distinguished name (cert.getSubjectDN())
- You might also read
  - http://java.sun.com/j2se/1.5.0/docs/guide/security/cert3.html

Managing Certs and SSL Sockets

- Use the KeyStore class for
  - Loading certificates from file (e.g., your CA certificate)
  - Storing programmatically generated certificates
- Use SSLContext class for setting up certificates to be used with SSLServerSocket
  - Create a certificate
  - Load into new KeyStore
  - Init a KeyManagerFactory with new KeyStore
  - Init SSLContext with new KeyManagerFactory and provided "TrustEveryone" TrustManager
- Use SSLContext for creating SSLSocketFactories
- See MITMSSLSocketFactory.java

Admin Server

- Already listens for client connections and parses the data sent using plain sockets
- You should
  - Modify code to use SSL sockets (see the proxy server code for examples)
  - Implement authentication for the transmitted username and password
  - Implement required admin commands
    - Shutdown
    - Stats

Password file

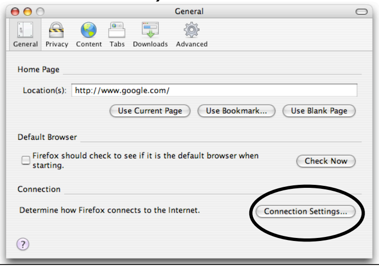
- Need to store a file containing usernames, salts, and hashed passwords
  - Both public and secret salts (aka pepper)
- Should be stored encrypted with an authenticated encryption scheme
  - I recommend Encrypt-then-MAC
  - Maybe AES in CTR mode to Encrypt, and HMAC-SHA1 to MAC
  - But be careful about security!!

Username	Salt	Hashed password
Alice	S	H(Pwd||S||P)
Bob	…	…

Password File Utility

- You should add a utility for creating these password files
- Simple method:
  - Make a class to take a file and a list of usernames and passwords, and covert it to a password file.

Configuring Firefox (under OS X, similar for Linux)


Home Page

Location(s): http://www.google.com/

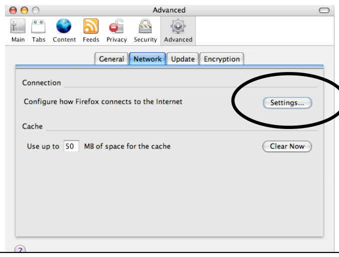
Default Browser

☐ Firefox should check to see if it is the default browser when starting.

Connection

Determine how Firefox connects to the Internet.

Configuring Firefox (under OS X, similar for Linux)

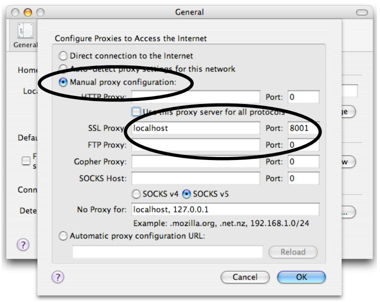


Connection

Configure how Firefox connects to the Internet

Cache

Use up to 50 MB of space for the cache



Configure Proxies to Access the Internet

- Direct connection to the Internet
- Auto-detect proxy settings for this network
- [✓ Manual proxy configuration:]

HTTP Proxy:		Port:	0
	Use this proxy server for all protocols		
SSL Proxy:	localhost	Port:	8001
FTP Proxy:		Port:	0
Gopher Proxy:		Port:	0
SOCKS Host:		Port:	0

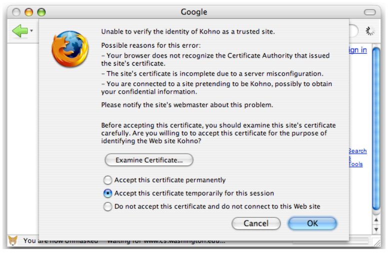
SOCKS v4 [✓ SOCKS v5]

No Proxy for: localhost, 127.0.0.1

Example: .mozilla.org, .net.nz, 192.168.1.0/24

- Automatic proxy configuration URL:

When going to https://www.cs.washington.edu



Unable to verify the identity of Kohno as a trusted site.

Possible reasons for this error:

- Your browser does not recognize the Certificate Authority that issued the site's certificate.

- The site's certificate is incomplete due to a server misconfiguration.

- You are connected to a site pretending to be Kohno, possibly to obtain your confidential information.

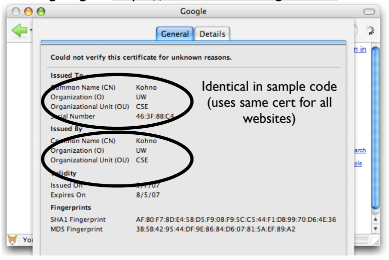
Please notify the site's webmaster about this problem.

Before accepting this certificate, you should examine this site's certificate carefully. Are you willing to to accept this certificate for the purpose of identifying the Web site Kohno?

- Accept this certificate permanently
- [✓ Accept this certificate temporarily for this session]
- Do not accept this certificate and do not connect to this Web site

You are now unmasked   Waiting for www.cs.washington.edu...

When going to https://www.cs.washington.edu

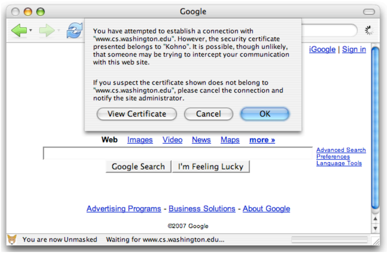


Could not verify this certificate for unknown reasons.

Issued To	
Common Name (CN)	Kohno
Organization (O)	UW
Organizational Unit (OU)	CSE
Serial Number	46:3F:8B:C4
Issued By	
Common Name (CN)	Kohno
Organization (O)	UW
Organizational Unit (OU)	CSE
Validity	
Issued On	5/7/07
Expires On	8/5/07
Fingerprints	
SHA1 Fingerprint	AF:B0:F7:8D:E4:58:D5:F9:08:F9:5C:C5:44:F1:DB:99:70:D6:4E:36
MD5 Fingerprint	3B:5B:42:95:44:DF:9E:86:84:D6:07:81:5A:EF:89:A2

Identical in sample code (uses same cert for all websites)

Sample code causes this second warning

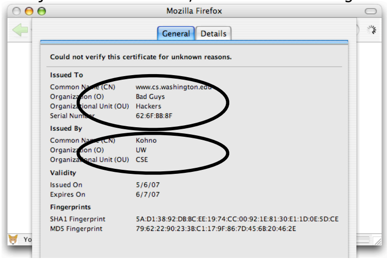


You have attempted to establish a connection with "www.cs.washington.edu". However, the security certificate presented belongs to "Kohno". It is possible, though unlikely, that someone may be trying to intercept your communication with this web site.

If you suspect the certificate shown does not belong to "www.cs.washington.edu", please cancel the connection and notify the site administrator.

You are now Unmasked   Waiting for www.cs.washington.edu...

Your job - new certificates, avoid second warning



Could not verify this certificate for unknown reasons.

Issued To

Common Name (CN)	www.cs.washington.edu
Organization (O)	Bad Guys
Organizational Unit (OU)	Hackers
Serial Number	62:6F:BB:8F

Issued By

Common Name (CN)	Kohno
Organization (O)	UW
Organizational Unit (OU)	CSE

Validity

Issued On	5/6/07
Expires On	6/7/07

Fingerprints

SHA1 Fingerprint	5A:D1:38:92:DB:BC:EE:19:74:CC:00:92:1E:81:30:E1:1D:0E:5D:CE
MD5 Fingerprint	79:62:22:90:23:3B:C1:17:9F:86:7D:45:6B:20:46:2E

Possible Problems

- You should be able to start up the proxy and connect to it "out of the box"
  - After you create your keystore with "keytool"
- If you are having problems
  - Is someone else trying to use your machine and that port? (Default 8001.)
    - Try a different port on the command line
  - Firewall problems
    - Try to telnet to the needed ports (8001/8002/...)
  - Try running your browser on the same machine, and setting the proxy as "localhost"
- Course mailing list: Great place to share knowledge