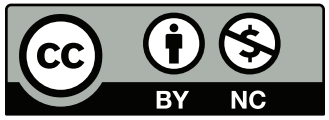
Authentication



With Whom Are You Communicating?

- Identification — who they claim to be
- Authentication — proof
- Authorization — what they can do

Current Focus

- Authentication
- Network-oriented (biometrics aren't much good over the net)
- Many different styles

Authentication

- Something you know (i.e., passwords)
- Something you have (i.e., some sort of token or smart card)
- Something you are (biometrics)

Shared Secrets

- Can be something you know or something you have
- Password
- Cryptographic key

Passwords

- Very guessable — 10-40%, according to several studies
- Forgettable — people write them down (though that's generally *not* a networked threat)
- Susceptible to eavesdropping on the network (often called *sniffing*, though that's actually the brand name of a very reputable network diagnostic device)
- Can be given away, social-engineered, phished, etc.

Guessing Passwords

- Explained well by Morris and Thompson (see the reading)
- Basic strategy: start with a good dictionary
- Match your dictionary to the target population: science fiction terms, other languages, etc.
- Include names: boyfriends, girlfriends, spouses, pets, etc.
- Try variations on this word list: add a period or a digit, change “o” to “0”, etc.

Eavesdropping

- Trivial on wireless networks
- Not hard on wired Ethernets, either, especially if the switch misbehaves (or can be induced to misbehave)
- Active attack: send out fake ARP responses, to direct traffic to you
- Best defense: encryption
- Some people use various one-time passsword schemes

Fooling an Ethernet Switch

- Switches learn what Ethernet (MAC) addresses live on which ports; they then send traffic for a given address only to the proper port
- Spoof your victim's MAC address; packets will go to you instead
- Send packets from *many* different (fake) MAC addresses, and exhaust the switch's MAC table — the switch will then flood all packets to all ports

Tokens

- Generally have an embedded shared secret used as a cryptographic key
- Time-based: encrypt the time of day (i.e., RSA's SecurID)
- Challenge/response: server sends a random number; token encrypts it
- These are *one-time passwords* — never reused
- (Database is updated to prevent immediate reuse with SecurID)
- Frequently used together with a PIN, to guard against device loss or theft
- Other advantages beyond eavesdropping protection: can't be social-engineered; if lent out, the authorized owner can't use them
- But — someone put his on a webcam: `http://fob.webhop.net/`


321812
RSA SecurID®

Active Attacks

- Simple one-time passwords aren't enough against active attackers
- Two attacks (depending on attacker's powers): last-digit guessing attack and connection hijacking

Connection Hijacking

- Use ARP-spoofing or equivalent to route traffic through you
- Wait for victim to log in
- Imitate victim; don't let packets through to real user
- Demonstrated in the lab in 1995; now doable with off-the-shelf hacker code
- Defense: cryptographically protect *all* packets

Last-Digit Guessing Attacks

- Watch user start to log in with SecurID (or equivalent)
- Open up ten simultaneous login sessions; as the user types each digit of the authenticator, repeat that on each of the new login sessions
- Before the user types the last digit, guess at each possibility, once per new session
- The attacker will win. . .
- Defense (other than crypto): lock the authentication database entry for that user after the login name is entered, rather than when the authenticator is received
- But — attacker can then start a login session and never send anything, as a denial of service attack. . .

Cryptography and Authentication

- Some way to use a cryptographic key to prove who you are
- (Much more on that next class)
- Can go beyond simple schemes given above
- Can use symmetric or public key schemes
- Most public key schemes use *certificates*

What are Certificates

- How does Alice get Bob's public key?
- What if the enemy tampers with the phone book? Sends the phone company a false change-of-key notice? Interferes with Alice's query to the phone book server?
- A certificate is a digitally-signed message containing an identity and a public key — prevents tampering.

Why Trust a Certificate?

- Who signed it? Why do you trust them?
- How do you know the public key of the *Certificate Authority* (CA)?
- Some public key (known as the *trust anchor*) must be provided out-of-band — trust has to start somewhere.

Certificate Authorities

- Who picks CAs? No one and every one.
- Your browser has some CAs built-in — sometimes because the CA paid the browser vendor enough money. Is that grounds for trust?
- Matt Blaze: “A commercial certificate authority can be trusted to protect you from anyone from whom they won’t take money.”

Who Gets Certificates?

- How do you prove your identity to a CA?
- How good a job do they do verifying it?
- What warranties does the CA give if someone is fooled? (Most disclaim all liability. . .)

Another Trust Model

- Get certificates from parties whom you know.
- Issue certificates to your own users: *authorization certificates*
- Don’t rely on commercial identity-based CAs.

Certificate Hierarchy versus Web of Trust

- Most CAs are tree-structured
- Top-level CAs can use *bridge CAs* to cross-certify each other
- PGP uses a different style: a *web of trust*.
- Certificate signings can form an arbitrarily-complex graph — users can verify path to as many trust anchors as they wish.

Styles of Certification

- At least 3 major styles
- X.509/PKIX — traditional hierarchical CA (but can have “pki” instead of “PKI”)
- SPKI/SDSI — authorization certificates
- PGP web of trust (primarily for email)

What Else is in a Certificate?

- Technical information, such as algorithm identifiers
- More identification information — company, location, etc.
- Expiration date
- Logos
- Certificate role

(What is X.509?)

- X.509 is an ITU standard for public key certificates
- On the Internet, most people use profiles defined by the PKIX (Public Key Infrastuture) working group of the IETF
- X.509 certificates are defined using ASN.1 (Abstract Syntax Notation 1) and encoded using DER (Distinguished Encoding Rules) or BER (Basic Encoding Rules)
- Use of libraries and tools for dealing with any of this is highly recommended!
- OpenSSL does most of what people need

Some Local Certificates

- CS dept web certificate at
  `http://www.cs.columbia.edu/~smb/classes/s09/cs-cert.txt`
- University web certificate at
  `http://www.cs.columbia.edu/~smb/classes/s09/cu-cert.txt`
- Signer's certificate for CU and CU-CS at
  `http://www.cs.columbia.edu/~smb/classes/s09/root-cert.txt`
- I retrieved those certificates via

  ```
  openssl s_client -connect www.columbia.edu:443
  ```

- Display certificates via

  ```
  openssl x509 -text <filename
  ```

Things to Notice About Certificates

- Signer (the university didn't issue the department's certificate)
- Validity dates (the signing certificate is good for 20 years)
- Algorithms (RSA, SHA1)
- Key length: 1024 bits for the university, 2048 bits for the department — but 1024 bits for the signer! (GeoTrust recommends 1024-bit keys.)
- Certificate usage — encryption and authentication, but *not* for issuing other certificates
- Certificate Revocation List (CRL)

Certificate Fields

- CU's Subject Name: `C=US, ST=New York, L=New York, O=Columbia University, OU=Information Technology, CN=www1.columbia.edu`
- CS Subject Name: `C=US, O=www.cs.columbia.edu, OU=564007148, OU=See www.geotrust.com/resources/cps (c)08, OU=Domain Control Validated - QuickSSL Premium(R), CN=www.cs.columbia.edu`
- The important field to a computer is `CN`: common name
- CRL: `URI:http://crl.geotrust.com/crls/secureca.crl`
- In the Equifax root certificate but not the others:

```
X509v3 Basic Constraints:
    CA:TRUE
```

Not All Certificates are Alike

- An email certificate isn't the same as an ecommerce certificate.
- A CA certificate is even more different — can I use my att.com email certificate to issue more att.com certificates?
- What about a code-signing certificate for ActiveX?
- Usage-specific information, such as IP address range
- The certificate type is listed in the certificate. Beyond that, end systems can apply their own policies, i.e., don't accept code-signing certificates from `www.evilhackerdudez.org`

Revoking Certificates

- Keys associated with certificates can be compromised
- One choice – *certificate revocation list* (CRL)
- Can get large, which is one reason why certificates expire
- For connected hosts, possible to do online certificate status checking
- Can the attacker block connectivity to the status server?
- CRLs are the weak link of public key cryptography.

Assurance

- Why should someone trust a certificate?
- Why should someone trust a root CA?
- See the *CPS* — certificate practice statement (but bring a lawyer. . .)

The GeoTrust CPS

4.5.2 "GeoTrust does not accept responsibility for reliance on a fraudulently obtained Certificate"

4.9.1 "In the event that GeoTrust ceases operation and there is no plan for transition . . . all Certificates issued by GeoTrust shall be revoked prior to the date. . . "

6.2.3 "The Root Keys for each CA Certificate are backed up but not escrowed."

9.2.3 "The GeoSure Protection Plan is an extended warranty program that provides certain GeoTrust certificate *subscribers* with protection against loss or damage. . . " [emphasis added]

9.6.4 "Relying Parties acknowledge that they have sufficient information to make an informed decision . . . to rely on the . . . Certificate"

PGP's Web of Trust

- Instead of buying a certificate from a commercial CA, have your friends sign your certificate.
- You, in turn, can sign theirs
- X.509 certificate topologies are trees, rooted at the CAs. Web of trust certificates form arbitrary directed graphs

Trust Anchors

- The starting point for evaluating a certificate is called a *trust anchor*
- With X.509, you start with a certain set of tree roots that you (or your software vendor) have decided to trust
- With PGP, you start your trust anywhere you want, dynamically, and try to assess the trustworthiness of the entire path
- *Lots* of keys out there for George W. Bush, of which none are valid. Who signed them?
- My key has about 40 signatures; if you trust *any* of those people, you can trust that my key is authentic
- Or — if you trust someone who signed the key of one of my signers, you can trust my certificate. (Note — you must trust honesty *and* competence.)

Authorization Certificates

- Issued by a site
- *Possession of the private key*, rather than the name, grants authorization
- Can still be revoked
- Does not require large, centralized PKI

"PKI" versus "pki"

PKI Centralized, generally outsourced, heavy-weight PKI to meet all possible needs, complete with giant CPS

pki Small, locally run, light-weight system, for granting access to local resources

- Do you want to trust (large-scale PKI provider) to identify your employees?

Does the Internet Need PKI? The Affirmative View

- Theoretically necessary
- Without it, how do you know if you're reaching the real Citibank?
- Active attacks are easy, and happen frequently; PKI provides server authentication

Does the Internet Need PKI? The Negative View

- Most end-users have no idea what a PKI is, what a certificate is, what a trust anchor is, or how to verify any of these.
- Most end-users don't know how well any given PKI provider verifies the identity of its subscribers
- Most users don't even notice if a given connection is encrypted, or the name in the URL bar, or anything else related to the authenticity of the connection
- It's security by assertion and magic

Actual Quote

> You can order our software right away using a secure server protected by secure socket layer (SSL) encryption.
>
> The share-it! order process is protected via a secure connection so that the data sent to the recipient can only be read by the recipient. Important information such as credit card numbers, addresses, etc. is sent to the recipient securely via the Internet.
>
> All of the data entered on the protected pages is encrypted using the SSL (Secure Socket Layer) protocol. Our servers support SSL Version 3 and 168-bit Triple DES encryption. The RSA module and SSL sessions feature 1024-bit encryption.

Impressive-sounding and true — but it ignores the threat to data on their disks.

Are PKIs Useful?

- PKIs *can* be useful
- More precisely, they're part of an overall security solution
- Their actual utility depends heavily on how they're used, how the software behaves, and how well-trained the user population is