

Services 3

Recall

Service -- protocol + port number

Telnet Trick

Text

Many services use a textual dialog between the client and server -- short phrases sent back and forth as lines of text.

Telnet

Telnet to the appropriate port number, and just type the commands by hand...

`% telnet host port`

"Transparency"

Systems are easier to debug if their state is visible and accessible. This is an advantage of text-based protocols.

ASCII standard

Using plain ASCII characters also provides portability. Every system in the world understands ASCII, and because each letter is a single byte, it does not depend on big-endian little-endian issues that make binary data harder to port between systems.

Telnet Examples

POP

(things typed by me are in bold)

`[localhost:~] nick% telnet pobox6.stanford.edu 110`

`Trying 171.64.14.86...`

`Connected to pobox6.stanford.edu.`

`Escape character is '^]'.`

`+OK pobox6.Stanford.EDU Cyrus POP3 v2.0.12 server ready`

`USER nick`

`+OK Name is a valid mailbox`

`PASS foobar`

`+OK Maildrop locked and ready`

`STAT`

`+OK 2 1242`

`+OK`

`QUIT`

`Connection closed by foreign host.`

HTTP

```
[localhost:~] nick% telnet www.stanford.edu 80
Trying 171.64.14.237...
Connected to www.lb-a.stanford.edu.
Escape character is '^]'.
HEAD / HTTP/1.0
```

```
HTTP/1.1 200 OK
Date: Mon, 23 Apr 2001 18:24:04 GMT
Server: Stronghold/2.4.2 Apache/1.3.6 C2NetEU/2412 (Unix)
  mod_fastcgi/2.2.4
Connection: close
Content-Type: text/html
```

Connection closed by foreign host.

--Security--

Most of our security coverage will come at the end of the quarter, but we'll do a little bit now to understand the security scheme in hw1b

Authentication

Prove who you are

3 ways

Something you know -- a password

Something you are -- fingerprint retinal scan (these methods have some serious disadvantages)

Something you have -- a little gadget which knows the password for you

Traditional Password

"Shared Secret"

The password is a secret thing known to both the client and server.

Password demand

The server asks the client for the password. The client provides it, the server checks it against the server's copy of the password.

One-Way Hash Function

Hash

Given a string as input, combine/hash the bits together in a regular way to produce a "hash" value.

Function of input

The hash value depends on every bit in the input. Changing the input a little results in a different hash value.

Not-invertible

Given the hash value, it should not be possible to compute the "inverse" -- figure out what the original input was to result in that hash. MD5 is a standard hash function.

Hash password DB

It's traditional for the server to not store the password directly. Instead, the server will store a hash of the password.

When the client sends over a password, the server hashes it, and compares the hash value to the stored value to see if they match.

Advantages

The server cannot impersonate the client.

Someone who breaks in to the server and steals the password db cannot impersonate the client.

Problem -- passwords in the clear

If the client sends the password to the server just as plain text, then a bad guy who was eavesdropping could pick up the password, and later impersonate the client.

Solution -- challenge/response

Challenge

The server sends a "challenge" to the client: a random number R.

Client

The client computes $\text{hash}(R+\text{password})$ and sends the hash to the server

Verify

The server knows R and knows the password, so the server can also compute $\text{hash}(R+\text{password})$ and see if it matches what the client sent.

Safe

Note that the bad guy can intercept R and $\text{hash}(R+\text{password})$, but in theory cannot invert the hash function to find the original password.

Problem -- password guessing

Bad guy

Knows R from observing the challenge

Knows hash(R+password) from observing the response

Cannot invert hash(R+password)

Can guess all possible passwords

Guess all possible passwords, and try hash(R+guess) for each one until a match is found to password(R+password)

Somewhat feasible

Say there are around 50,000 words ($5e4$) in the English language

Combinations of two words = $5e4 * 5e4 = 2.5 e9 = 2.5$ billion combinations

A large number, but not infeasible

Put the odds on your side with random sequence of letters, digits, punctuation

I use nonsense words with junk chars added in

Encrypted Connection

Another solution would be to bring up the connection between the client and server so that all the traffic on it is encrypted. Then the bad guy cannot intercept anything, and we could go back to just sending the password.

We'll study that technique later.

Password guessing is still a problem if the bad guy obtains the password database.

--Standards Power--

(some of this material is detailed on the previous handouts)

So Far

TCP/IP, POP, HTTP, SMTP, FTP, ...

What makes a standard

Public

Freely implementable

Observe

Protocols based on open standards keep dominating the vendor specific versions. Again, and again and again.

How?

How does that keep hapening? Tech merit of the standards?