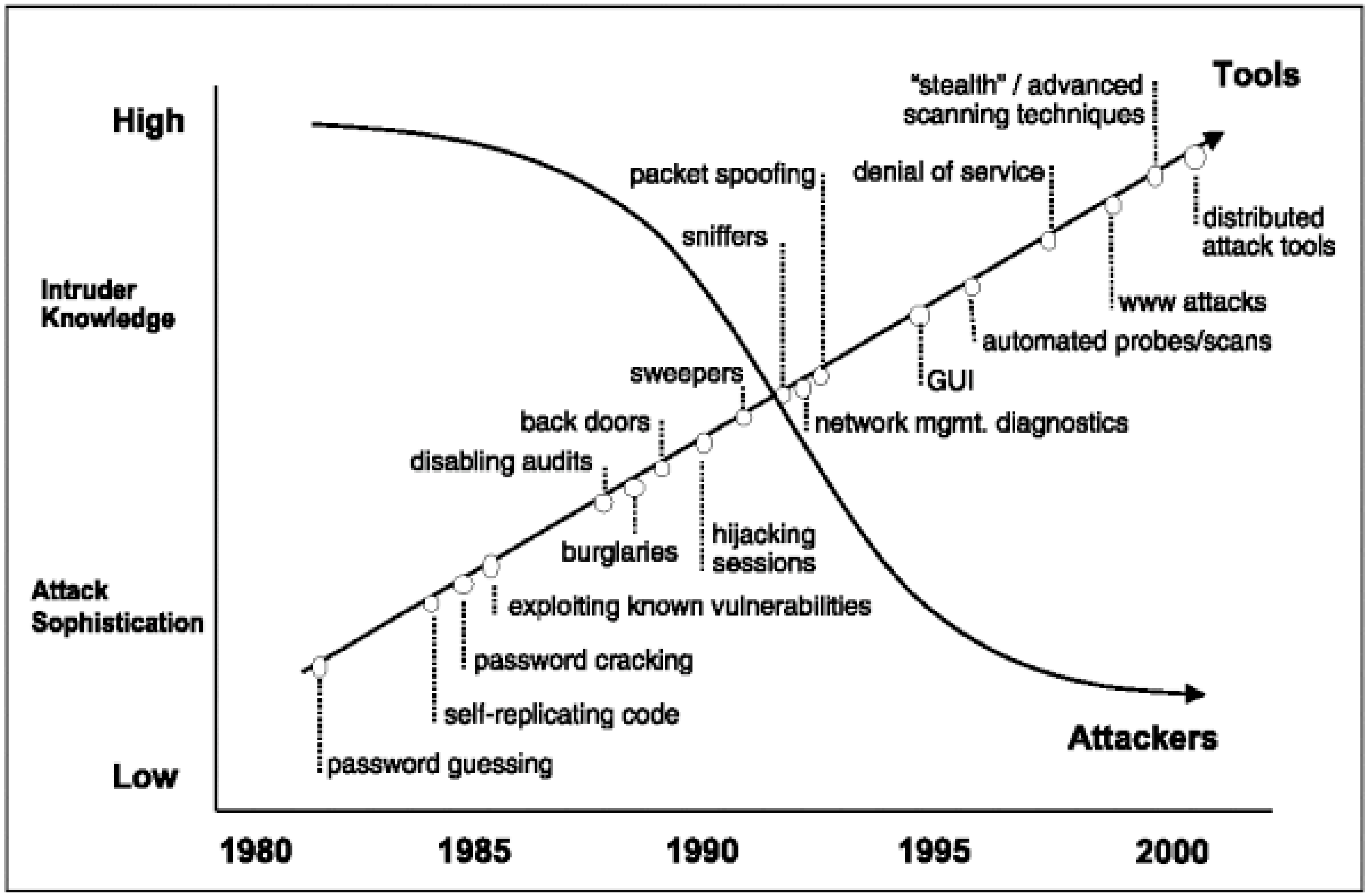
Security Applications

Computer Networks 15-441
Spring 2006
Joshua Hailpern

Evolution of Tools &Attacks

Figure 1. *Attack Sophistication vs. Intruder Technical Knowledge*



Who to bite?

- **Is a host alive?**
  - **Use `ping` (ICMP ECHO request and reply)**
- **Is a host running, say, a `telnet` server?**
  - » **Port scan (most servers listen on well-known ports)**
    - – **TCP: try `connect()` on all ports (`ECONNREFUSED`)**
    - – **UDP: try `sendto()` on all ports (`ICMP_UNREACH_PORT`)**
  - » **"Stealth scan"**
    - – **E.g. `nmap` (www.insecure.org)**
- **What OS is a host running?**
  - » **Different OS reacts differently to special packets**

Popular Port Scanning

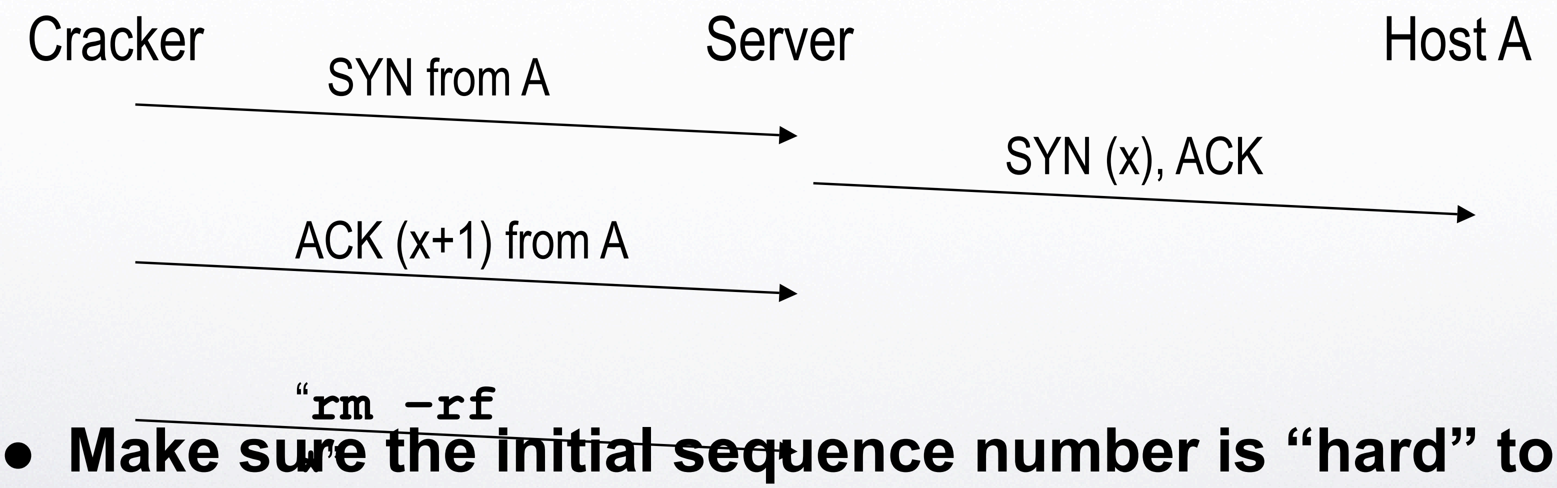
- **NMAP** - *http://www.insecure.org/nmap*
  - TCP scans - connect to every port with 3-way handshake
  - UDP scans
  - SYN scans using IP fragments
  - ACK and FIN scans
  - *designed to by-pass firewalls and intrusion detection tools*
- **QueSO** - http://www.apostols.org/projectz/queso
  - TCP scans with various combinations of TCP flags: SYN, SYN+ACK,FIN, FIN+ACK,SYN+FIN

Gain Access

- Direct access
  - Backdoor
  - Use the passwords obtained from packet sniffing
  - Password guessing
    - E.g. use “dictionary attack” on `/etc/password`
  - Bribery, blackmail, torture, etc.
- Exploit vulnerability to gain access
  - Protocol vulnerability
    - E.g. TCP sequence number prediction
  - Software vulnerability
    - E.g. buffer overflow, format string, etc.

Sequence # Prediction

- **Problem if a server uses IP/hostname based authentication**
    - **E.g. "**`.rhost`**" for** `rlogin`



- **Make sure the initial sequence number is "hard" to predict**
- **(Note: the cracker is also doing "spoofing")**

Session Hijacking

- *Allows an attacker to steal, share, terminate, monitor and* log any terminal session that is in progress
- *Session stolen across the network*
- What can be hijacked:

    - telnet , rlogin , rsh , ftp

  •Simple Session hijacking scenario:

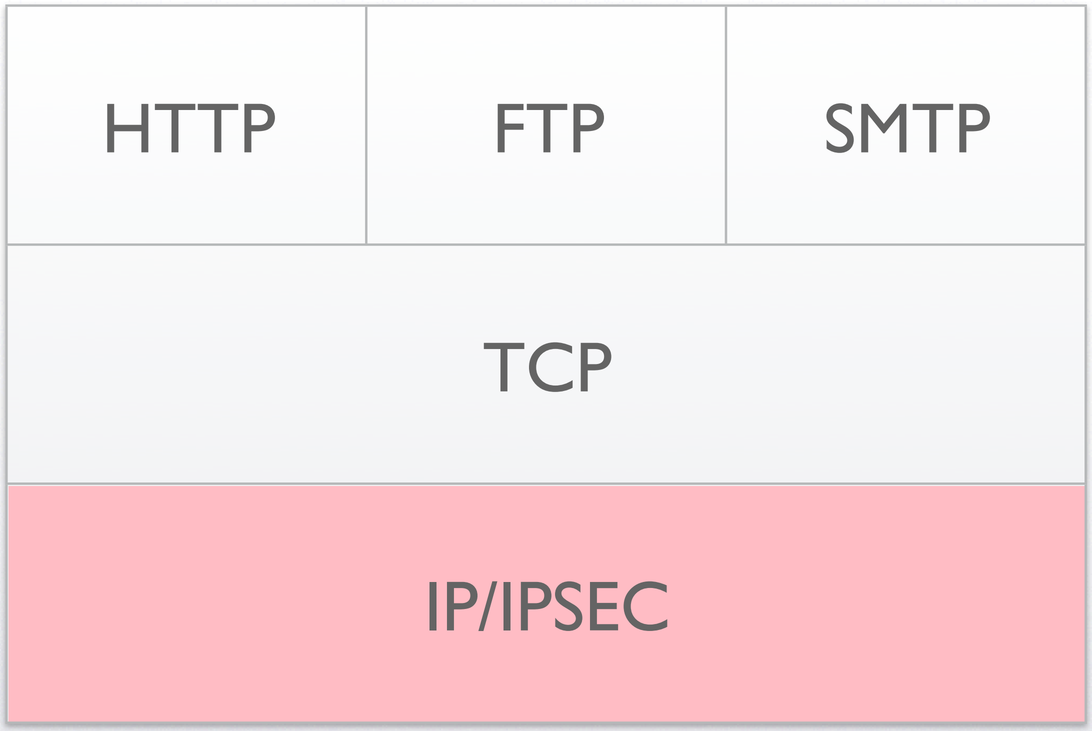
  – A telnets to B to get some work done

– Attacker resets connection to A

– Attacker kicks off A and takes over the session to B.

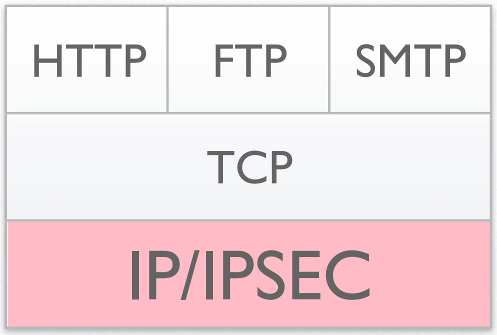
Levels of Security

Network Level



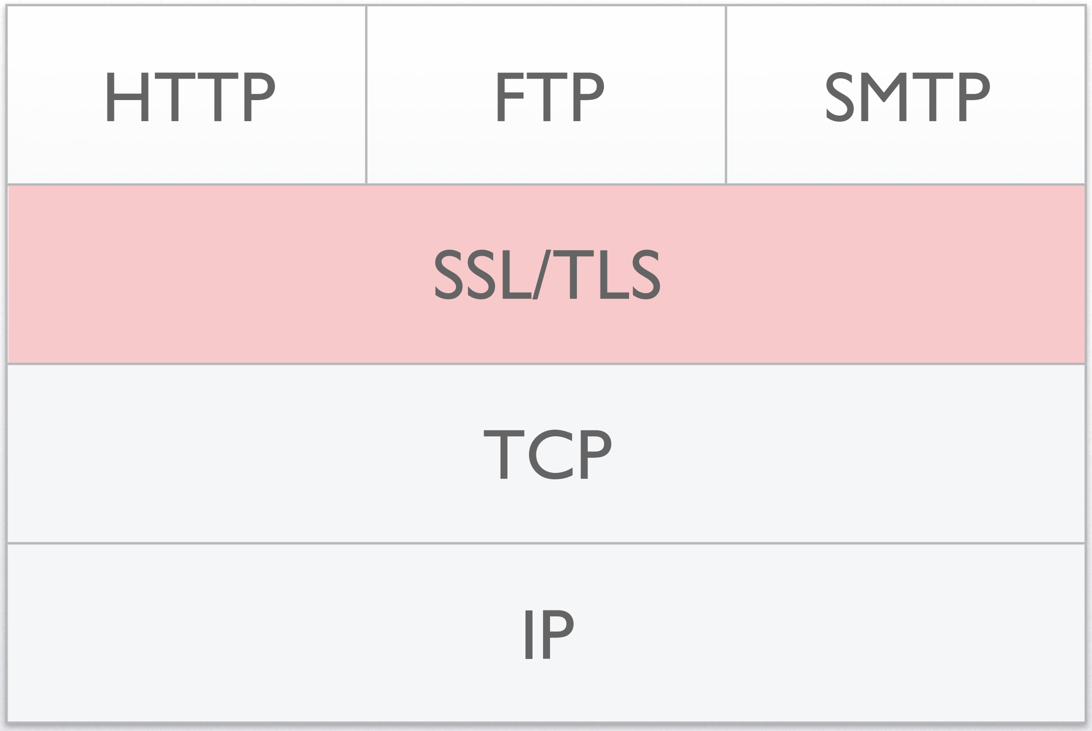
Network Level

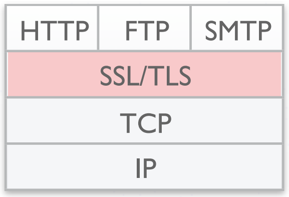
- protocol suite & tools
- encrypt & authenticate packets
- Virtual Private Network (VPN)
- RFC 2401



Levels of Security

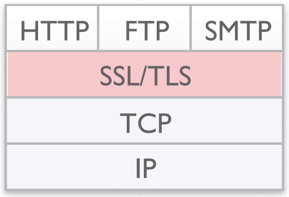
Transport Level





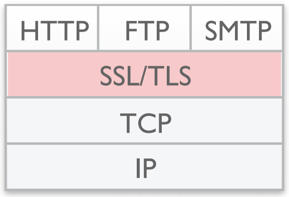
Network Level

- **Key concept: certificate**
- **Plan (conceptual)**
  - **Contact a server you suspect is www.FJALJFDSL.org**
  - **It will send you a certificate containing its public key**
  - **You will generate a random symmetric-cipher session key and encrypt it with the server's public key**
  - **Only www.FJALJFDSL.org can decrypt the message and obtain the session key**



Network Level

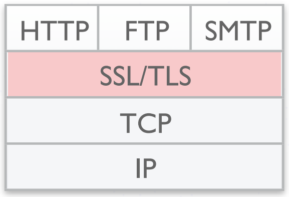
- **Is this ok?**



Network Level

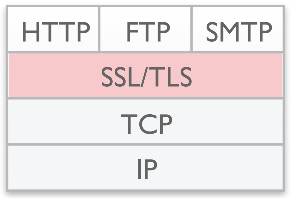
SSL

- **Trust?**
  - **No Global Authority!**
- **Solution!**
  - **“Trustworthy” organizations. (ie. Baltimore CyberTrust)**
- **Problems?**
  - **Server could provide fake certificate**
  - **How can you trust certificate providers?**
- **Solution!**
  - **Indirect Verification**
  - **Server provides www.FJALJFDSL.org certificate, and CyberTrust certificate.**



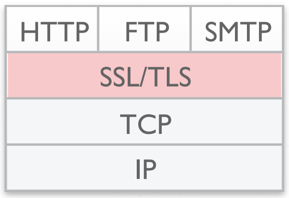
Network Level

- **Key concept: _signed_ certificate**
  - **To whom it may concern, the private key matching public key 2398898ca76fe676bbabe67867d00d7987bad is held by the owner of www.FJALJFDSL.org.**
    - **Sincerely, Baltimore Cybertrust**
  - **Hash: 469341329473a6755e5f5675a65b**
  - **Signature: 5fe65765865ca765b58675e5655a65c567586e65**
- **What could go wrong?**



Network Level

- **What could go wrong?**
  - **Maybe Baltimore CyberTrust didn't claim exactly that (maybe the domain name was different, maybe the key was different...)**
    - **Server could provide bogus certificate**
  - **Who is Baltimore CyberTrust anyway?**
    - **How do I know *their* public key?**
    - **How do I know *they* aren't crooks?**
- **One approach – insert a level of indirection**
  - **Server provides www.FJALJFDSL.org certificate**
  - **Server also provides Baltimore CyberTrust certificate**
    - **"To whom it may concern, the private key matching public key ... is held by the owner of Baltimore CyberTrust...Signed, ReallyTrustworthyPeople."**
- **"Certificate Chain"**



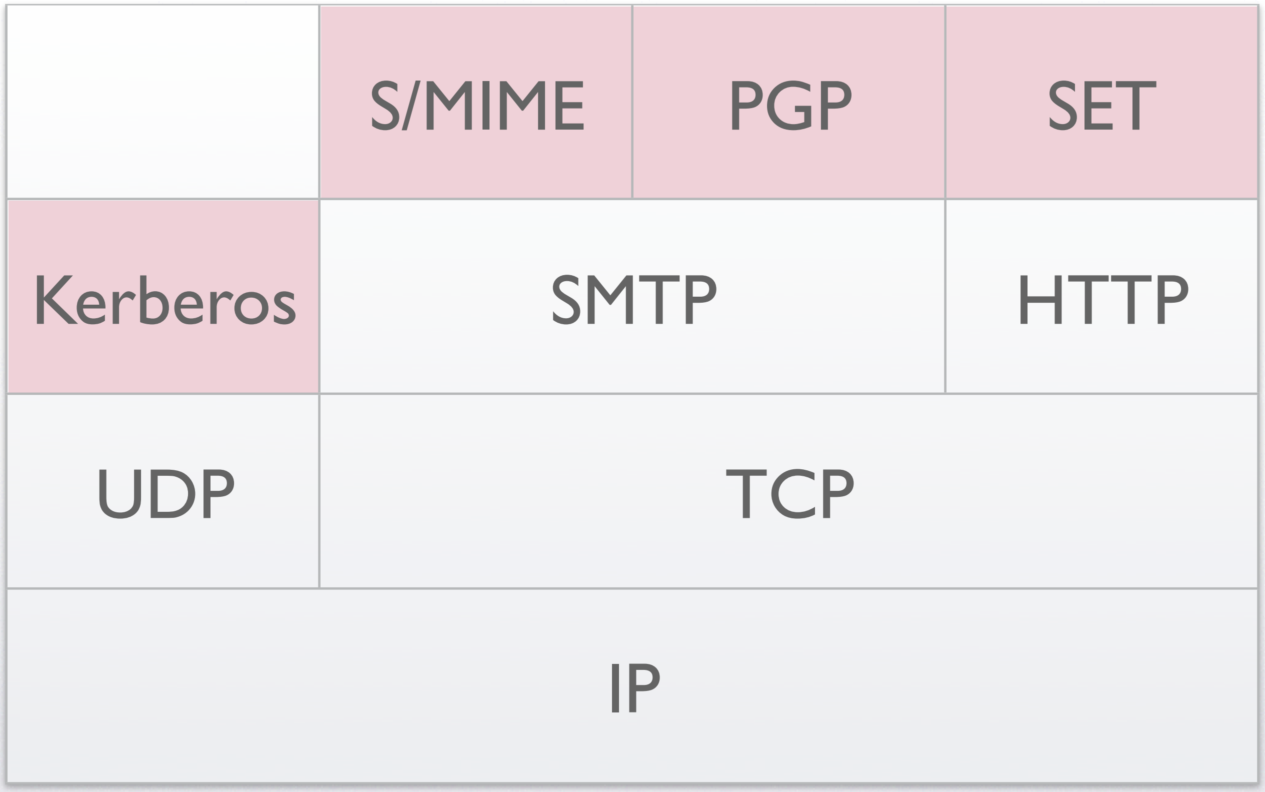
Network Level

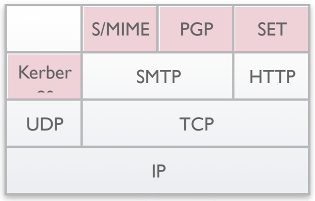
SSL

- **What does this mean?**
  - **Every browser has a list of CA**
  - **You can edit that list.**
  - **CMU has a CA!**
  - **Only “security” is by hand scanning and removal.**

Levels of Security

Application Level





Application Level

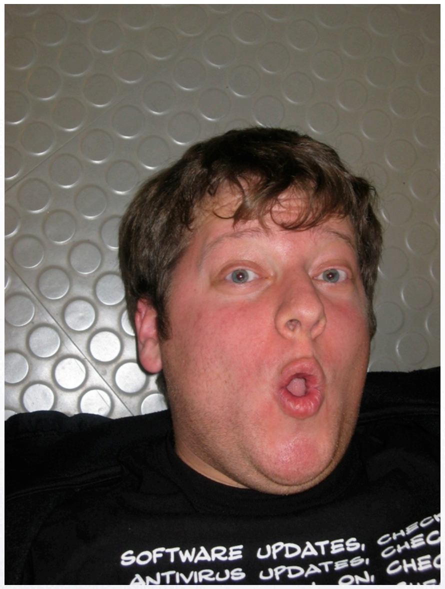
Kerberos

- **Uses symmetric cryptosystem (DES).**
  - **Key derived by one-way function from user's password.**
- **Kerberos 5 is an Internet Standard.**
- **Kerberos is an example of a centralized key distribution center.**
  - **Performance of private key cryptography without need to maintain N^2 key pairs**
  - **Every user shares a private key with a key distribution center**
    - **Called a Kerberos Authentication Server (AS)**
  - **When Bob and Alice want to communicate securely, Bob requests a one time (shared) session key from the KDC**
  - **The session key is distributed only to Bob and Alice**

	S/MIME	PGP	SET
Kerberos	SMTP		HTTP
UDP	TCP		
IP			

Application Level

Kerberos

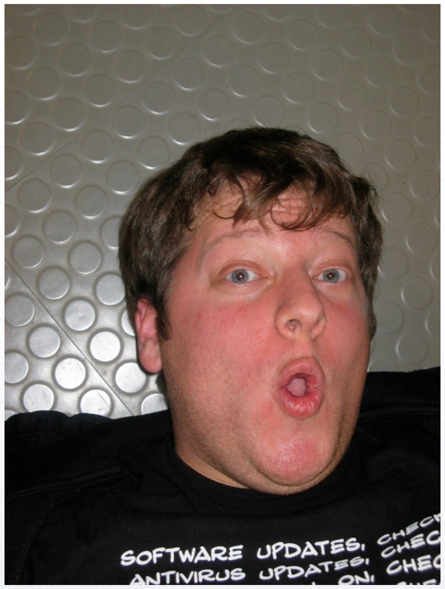


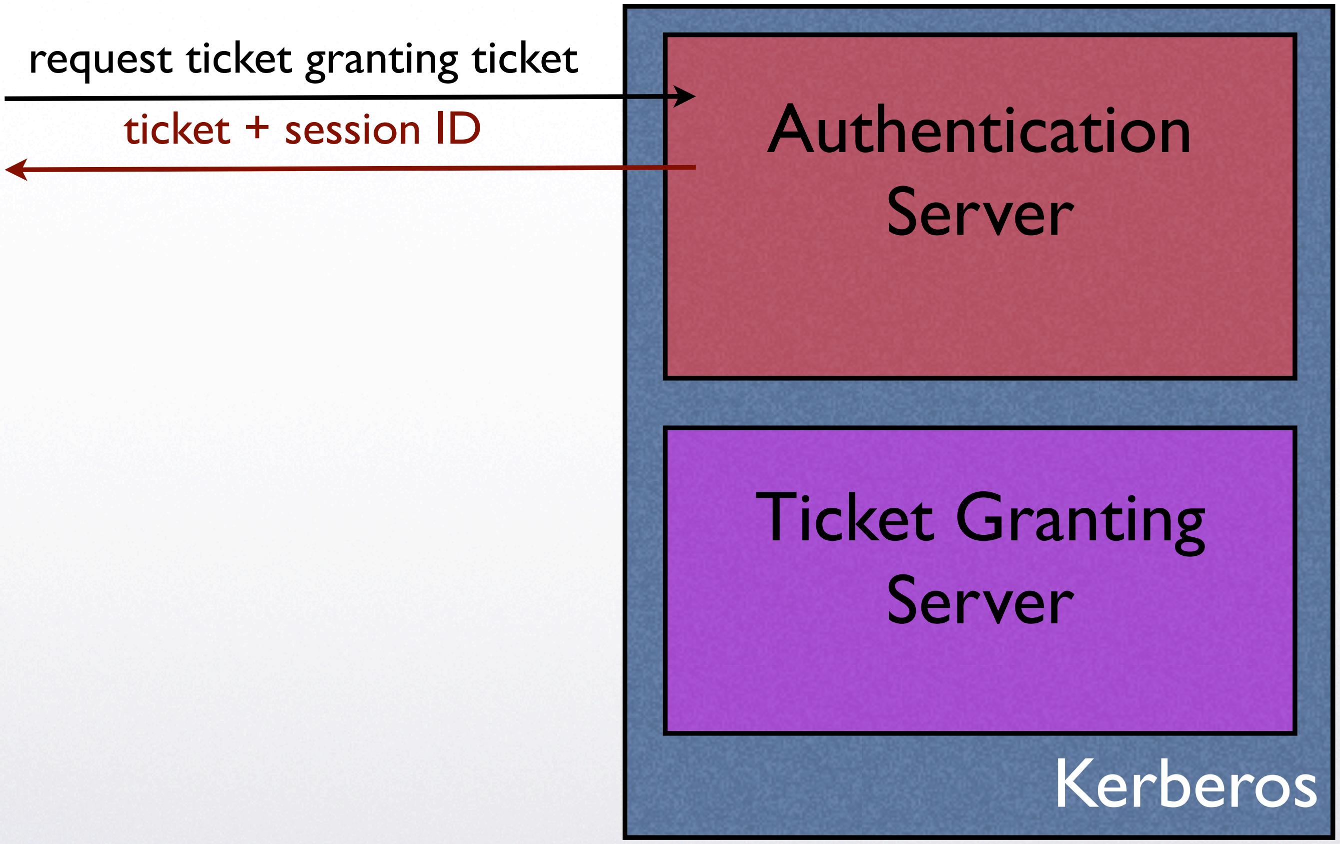
Bob Logs into Workstation & requests service on host

	S/MIME	PGP	SET
Kerberos	SMTP		HTTP
UDP	TCP		
IP			

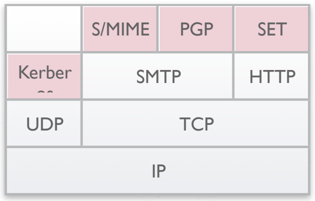
Application Level

Kerberos



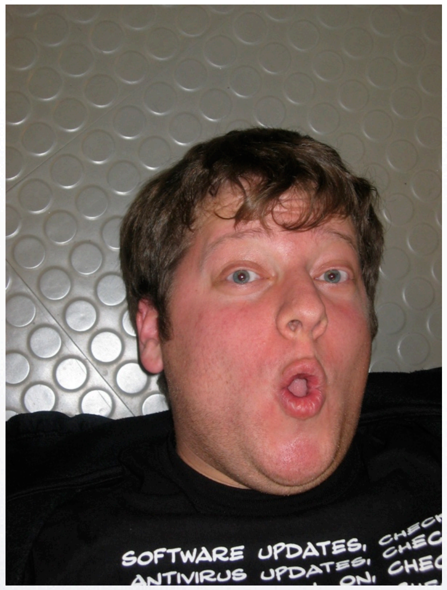


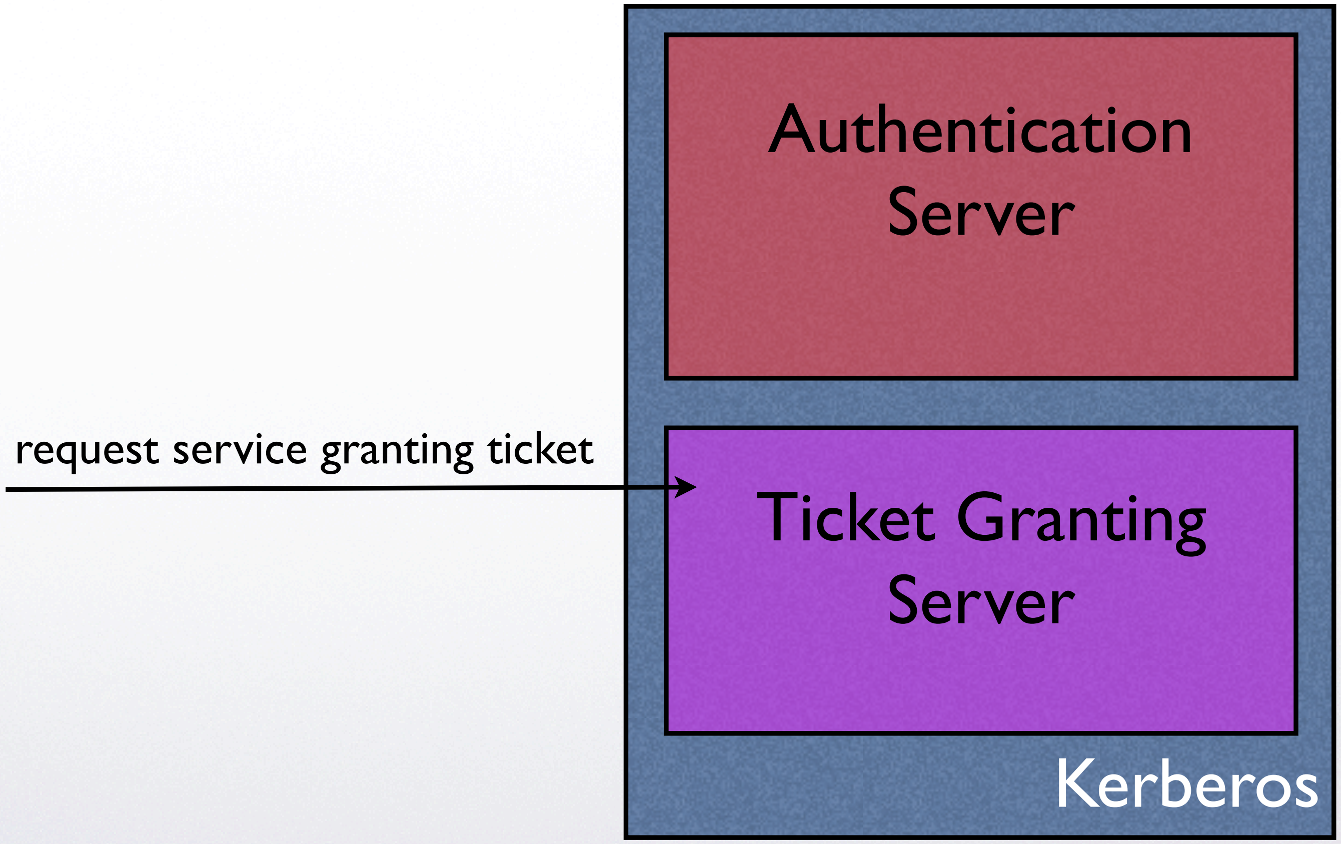
AS verifies access rights, creates ticket-grating ticket and session key.
Results encrypted using key derived from user's password


S/MIME
PGP
SET
Kerber os
SMTP
HTTP
UDP
TCP
IP

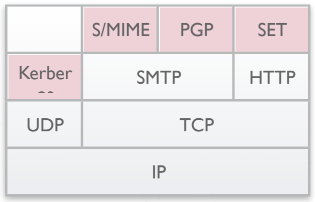
Application Level

Kerberos



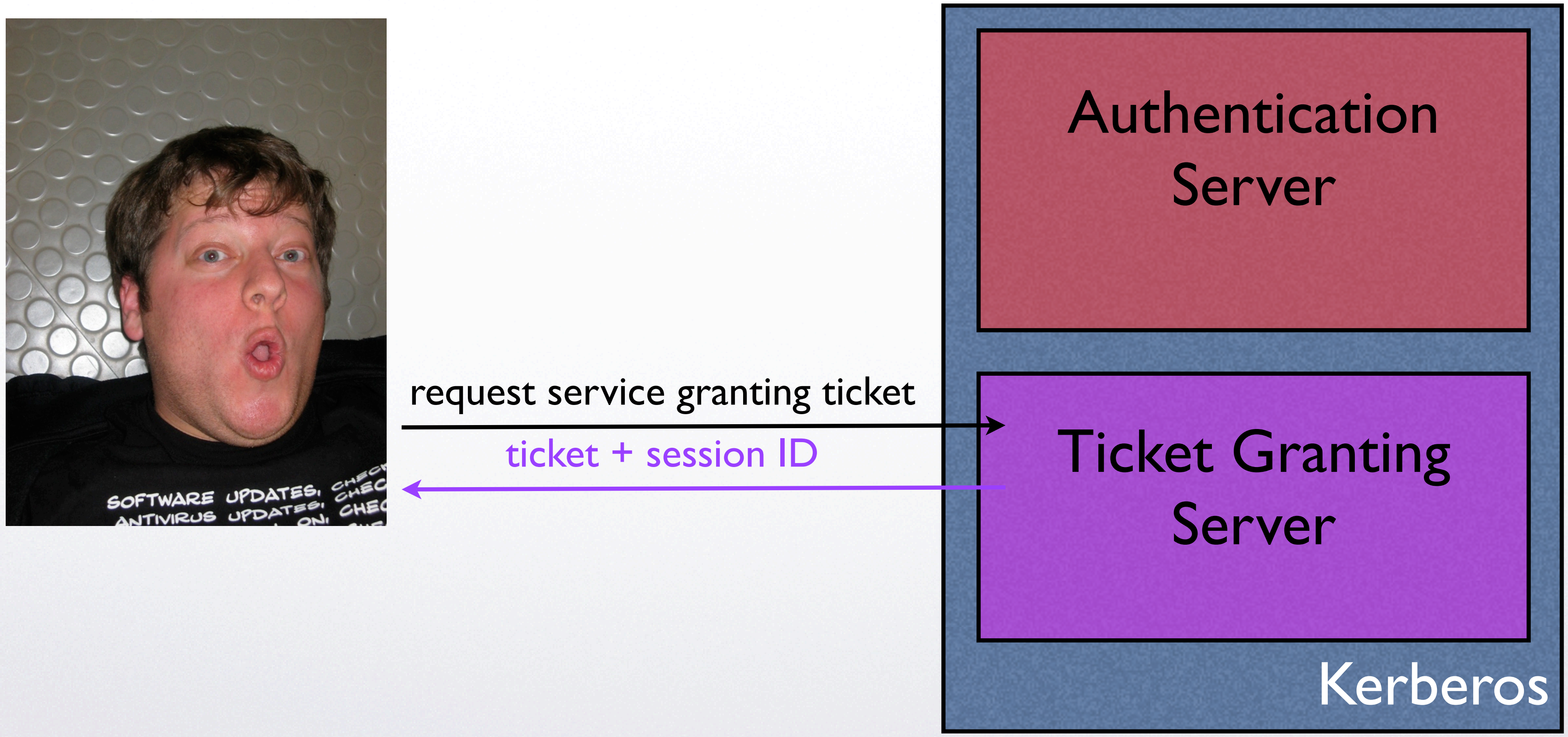


Workstation prompts Bob for PW, then decrypts. Sends Tickets and authenticator (containing user name and address) to TGS

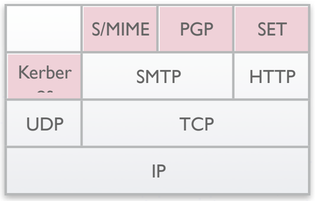


Application Level

Kerberos

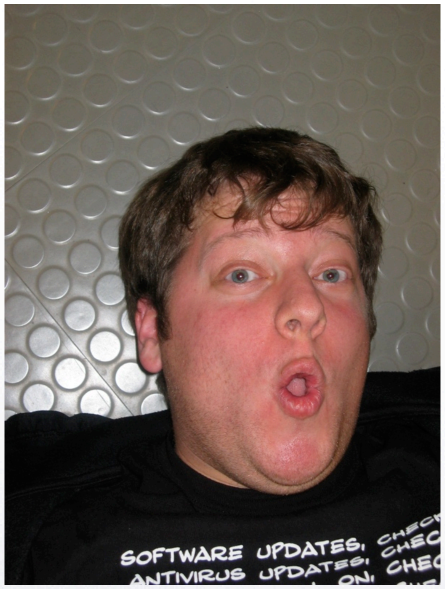


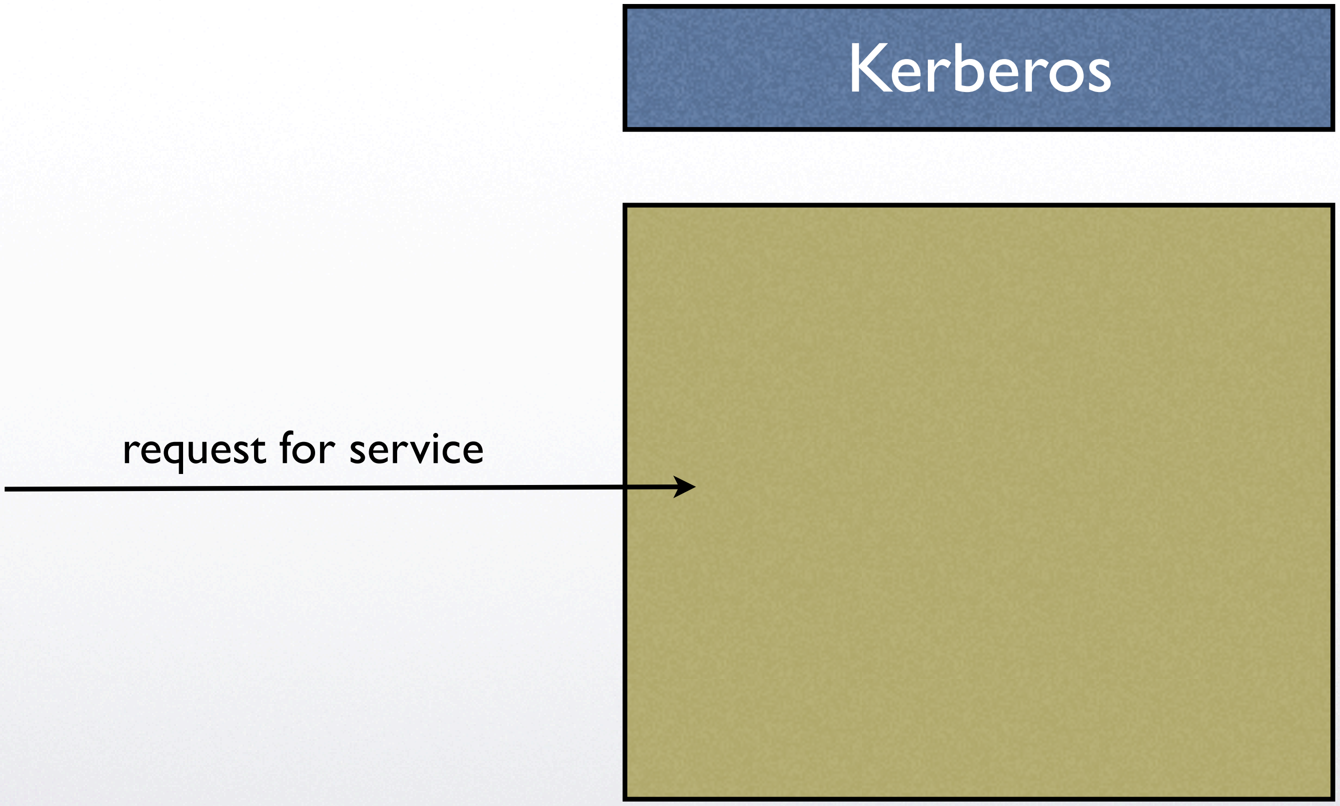
TGS decrypts ticket and authenticator verifies request then creates ticket for requested server.



Application Level

Kerberos



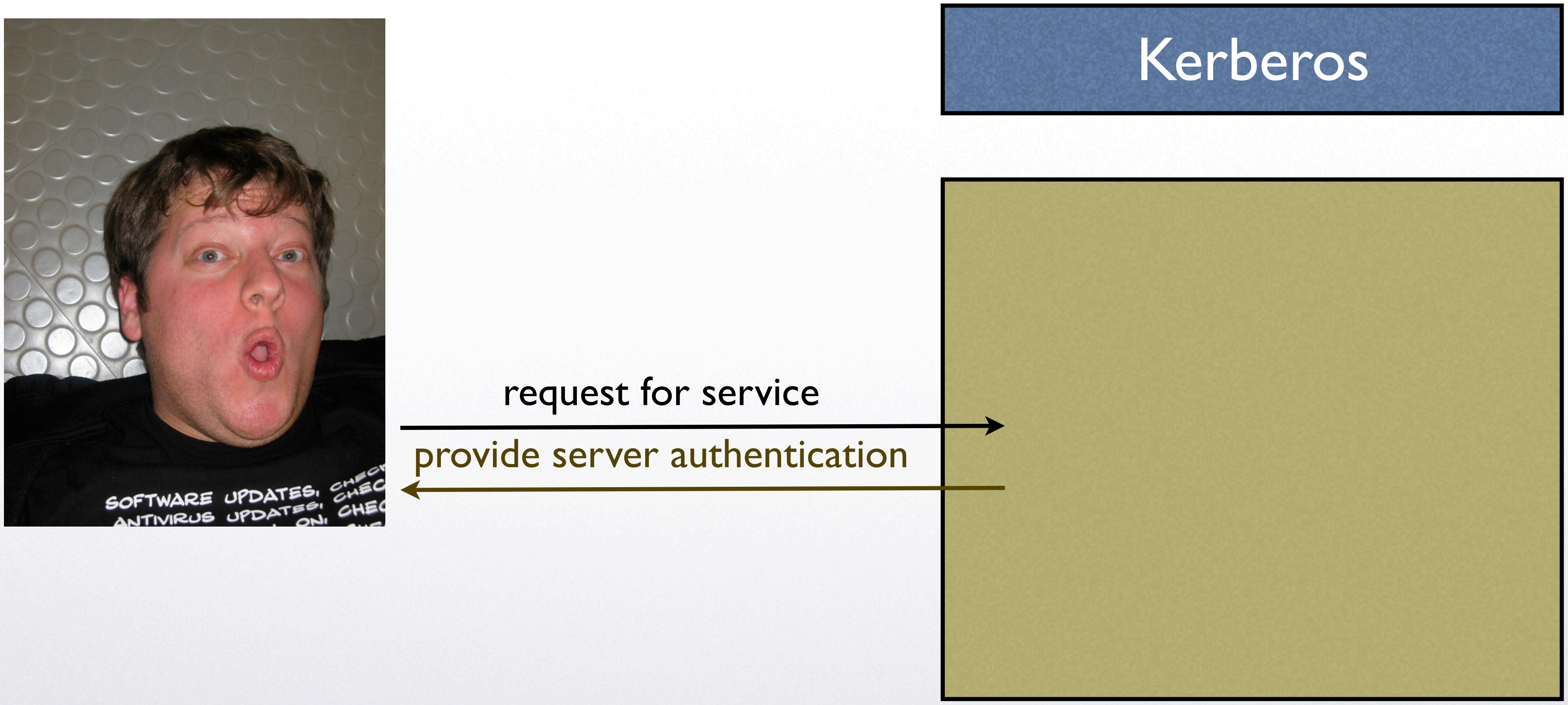


Workstation sends ticket and authenticator to server.

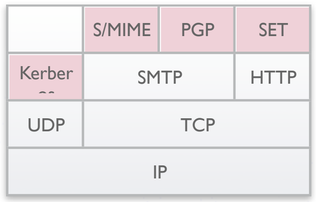
	S/MIME	PGP	SET
Kerberos	SMTP		HTTP
UDP	TCP		
IP			

Application Level

Kerberos



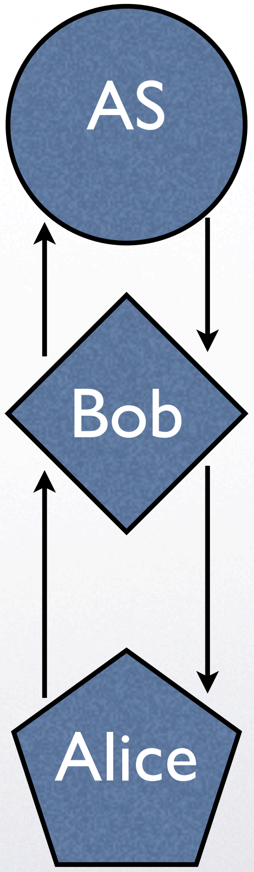
Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns authenicator


	S/MIME	PGP	SET
Kerberos	SMTP		HTTP
UDP	TCP		
IP			

Application Level

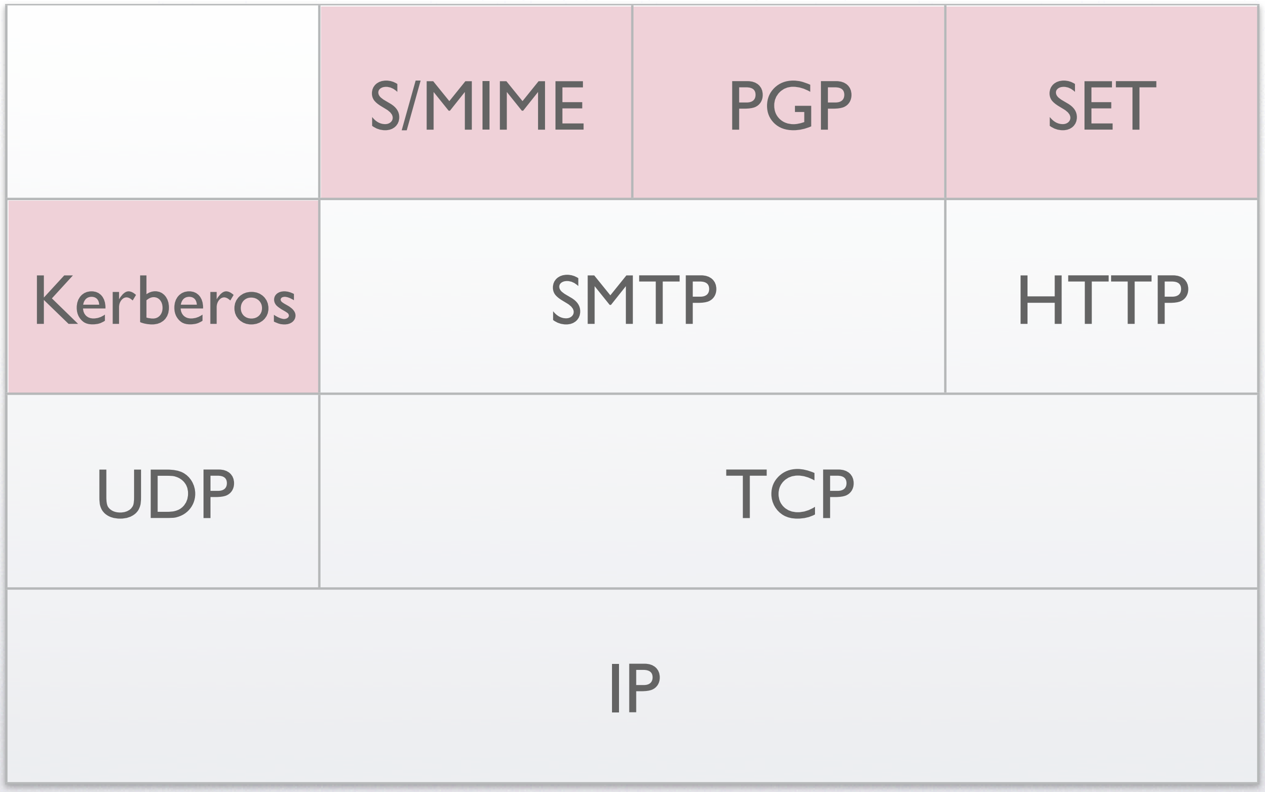
Kerberos

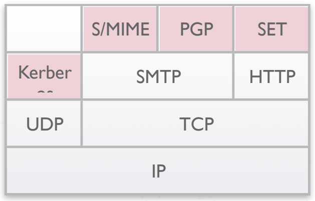
- **Bob tells AS that he wants to talk to Alice.**
  - **Encrypted using Bob's private key**
- **AS authenticates Bob, checks he has access privileges for Alice, and generates a session key for communication between Bob and Alice.**
- **AS generates a ticket intended for Alice.**
  - **Bob's name, the session key, and a timestamp**
  - **The ticket is encrypted using Alice's private key**
- **AS sends Bob the ticket plus session key.**
  - **Encrypted using Bob's key**
- **Bob then contacts Alice with the ticket plus an encrypted timestamp.**
  - **Alice decrypts the ticket, plus timestamp and sends back the timestamp plus one (nonce)**



Levels of Security

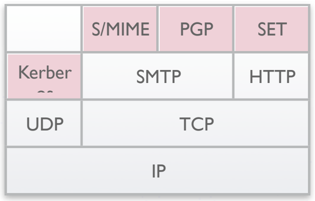
Application Level





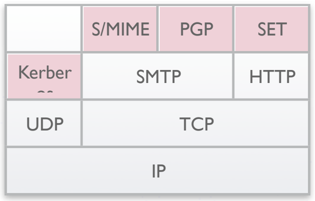
PGP

- **"Trust no one"**
  - **Why should I trust VeriSign, RSA, or any of the Certification Authorities?**
- **The PGP approach: "web of trust"**
  - **If I believe a key is really Bob's public key (e.g. get a disk from Bob), then I digitally sign the key to certify it**
  - **If I "trust" Mulder, and Mulder digitally signed Alice's public key, then I will believe the key is really Alice's public key**
    - **Assume I have Mulder's public key, so I can verify his signature**
- **Of course, you may think, "why bother?"**
  - **If I get Bob's public key from his web page, it's "probably" his**



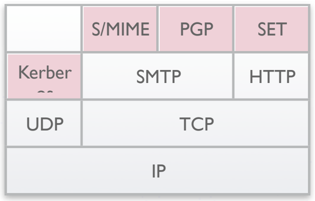
PGP

- Pretty Good Privacy
- Send secure emails
- Concept
  - Encrypt(Y_{public}, MSG)
  - "sign" E($M_{private}$, MSG)
  - E($M_{private}$, hash(MSG))
- Problems?



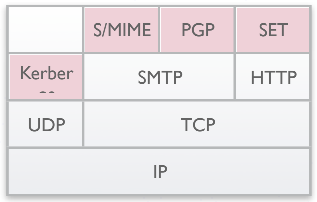
PGP

- **HUGE chunks of information**
- **VERY SLOW (because of large keys)**
- **how can we improve?**



PGP

- DES(KEY$_{session}$, MSG)
- Encrypt(Y$_{public}$, KEY$_{session}$)
  - <for multiple people>
  - Encrypt(X$_{public}$, KEY$_{session}$)



PGP

- **Encrypt($M_{private}$, Hash(MSG))**
- **Symetric Cypther($KEY_{session}$, MSG)**
- **Encrypt(Y_{public}, $KEY_{session}$)**
  - **<for multiple people>**
  - **Encrypt(X_{public}, $KEY_{session}$)**

- **Now if Alice sends to Bob, Bob can verify it came from Alice, using the hash, and Alice can send it with ease to many people, using fast cypher, and small keys!**

Breaking into hosts

- Guessing passwords
- Port scans, …
- Stack overflow
- TCP hijacking, SYN attack

Breaking Into Hosts

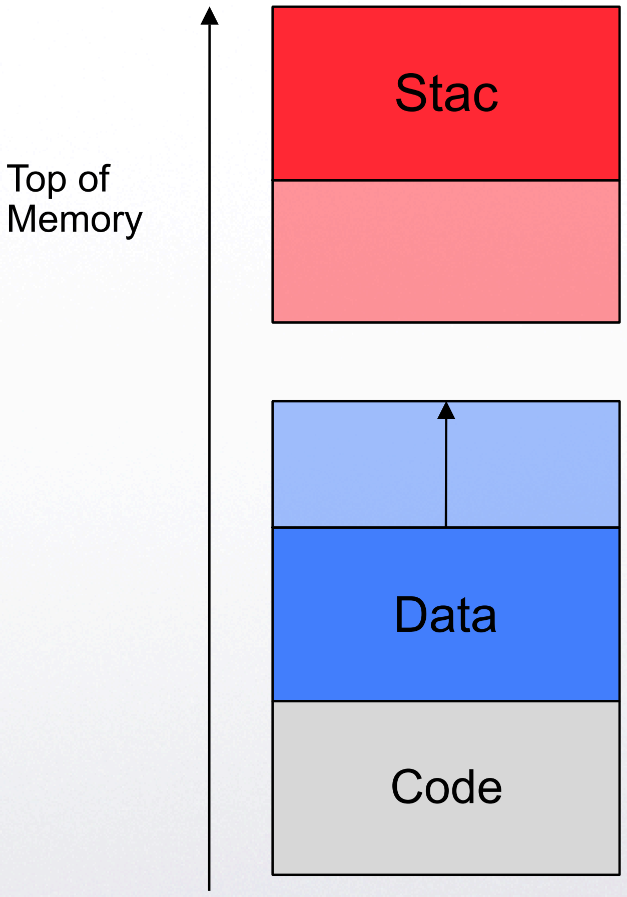
- Guessing passwords
- Port scans, …
- Stack overflow
- TCP hijacking, SYN attack

Buffer Overflow Attacks

- One of the most used “hacking“ techniques
- Advantages
  - Very effective
    - attack code runs with privileges of exploited process
  - can be exploited locally and remotely
  - interesting for network services
- Disadvantages
  - Architecture/OS Version dependent
    - directly inject assembler code / call system functions
  - some guess work involved (correct addresses)

Process Structure

- **Three memory areas**
  - **Stack segment**
    - **local variables**
    - **procedure calls**
  - **Data segment**
    - **static (global) variables (bss)**
    - **dynamic variables (heap)**
  - **Code/Text segment**
    - **program instructions**
    - **usually read-only**



Stack Overflow Attack

- Data is copied into local variables without proper bound checking
  - vulnerable functions: *strcpy, strcat, gets, fgets, sprintf ..*
- Data “overflows“ allocated buffer and overwrites stack data (especially return address)
  - If done with a random data, this usually creates a segmentation fault
- Carefully overwrite content and set return address to user defined value
  - causes a jump to user defined code – modify execution flow
  - have this code executed with privileges of running process

Stack Overflow: Code

- **What code should be placed in the buffer?**
  - » **Assembly instructions, system calls, alignment, ..**
  - » **Different variations for different platforms**
  - » **Do not know addresses**
- **Usually, a shell is started.**
  - » **Use system call (execve) to spawn shell**
  - » **Linux system calls are invoked by passing arguments on the stack or in registers and calling 0x80 interrupt**
  - » **Runs with same priviledges as application**

After Gaining Access

- Obtain confidential information
  - E.g. emails, credit card numbers, etc.
- Destroy files, prevent login, …
- Use the host as a base for future attacks
  - Use it for a DDoS attack
  - Use it to gain access to other machines in a corporate network
  - Install "rootkit": modified system tools, for example:
    - `ps`: won't display certain processes
    - `ls`: won't display certain files
    - `netstat`: won't display certain network connections
  - Run packet sniffer to obtain more information (e.g. passwords)
  - …

A Social Engineering Attack

- An attempt by a computer hacker to persuade a legitimate system user to reveal information.
- Most common way hackers break into systems
- "If you give me your logon ID and password, I can fix it in a few minutes, you can change your password when I am done"…..
  - *A real help desk person will never ask you this !!*
- Hacker takes advantage of the organization size - people do not know each other.
- Ignorance is a big help (to the attacker)!

Denial of Service Attacks

- Make services unavailable.
- Typically achieved by wasting resources associated with the service.
  - Network bandwidth, memory, CPU cycles
  - Challenge: make the defense cheap
- Common attacks.
  - SYN attack, SMURF, ..
- IP traceback.

Denial of Service (DoS)

- There are countless DoS attacks out there today

ftp://info.cert.org/pub/tech_tips/denial_of_service

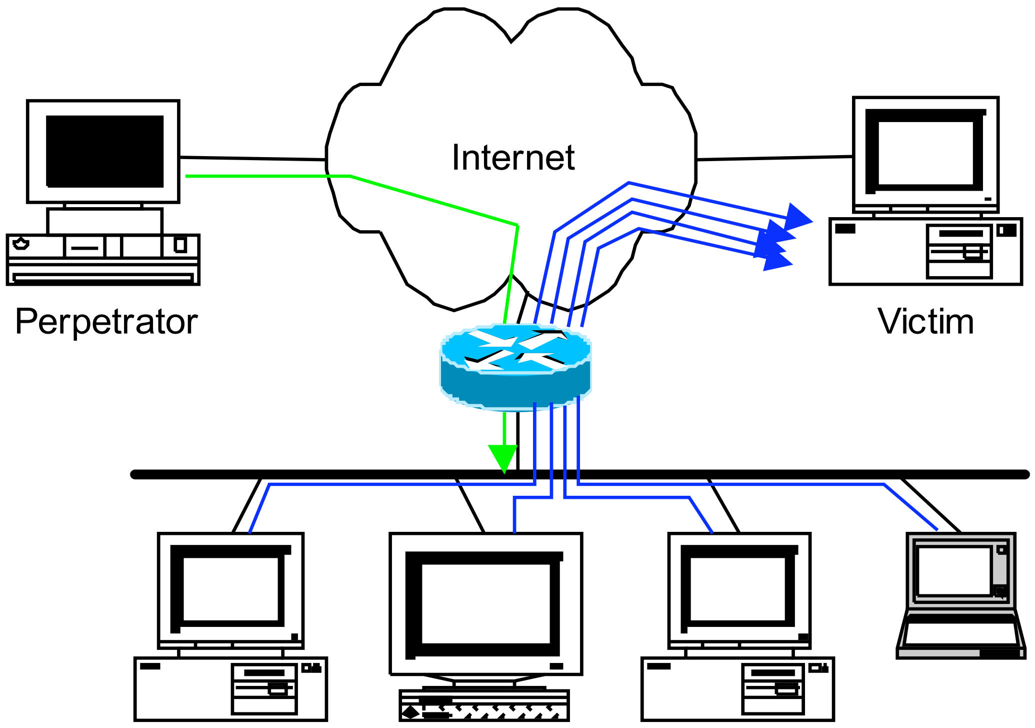
- Various forms:
  - SYN Flooding
  - Land (and similar)
  - Teardrop (and similar)
  - Smurf, papasmurf
  - Ping of Death

DOS: TCP SYN Flooding

- TCP is subject to SYN Flooding
- TCP based on 3-way handshake *(ISN - initial sequence number)*
    - *A ------SYN(A,ISN$_A$)----------------------->B*
    - *A <----ACK(A,ISN$_A$),SYN(B,ISN$_B$)-------B*
    - *A ------ACK(B,ISN$_B$)------------------------>B*
- Systems must allocate resources for each SYN to come in
- SYN attack scenario
    - Attacker sends several SYN packets to a victim from a spoofed (fake) machine SYN(X,ISN$_X$).
    - Connection cannot be ACK'd and waits for timeout.
    - The queue will fill up and the machine can go down or does not serve more requests.

SMURF

🟩	ICMP echo (spoofed source address of victim) Sent to IP broadcast address
🟦	ICMP echo reply

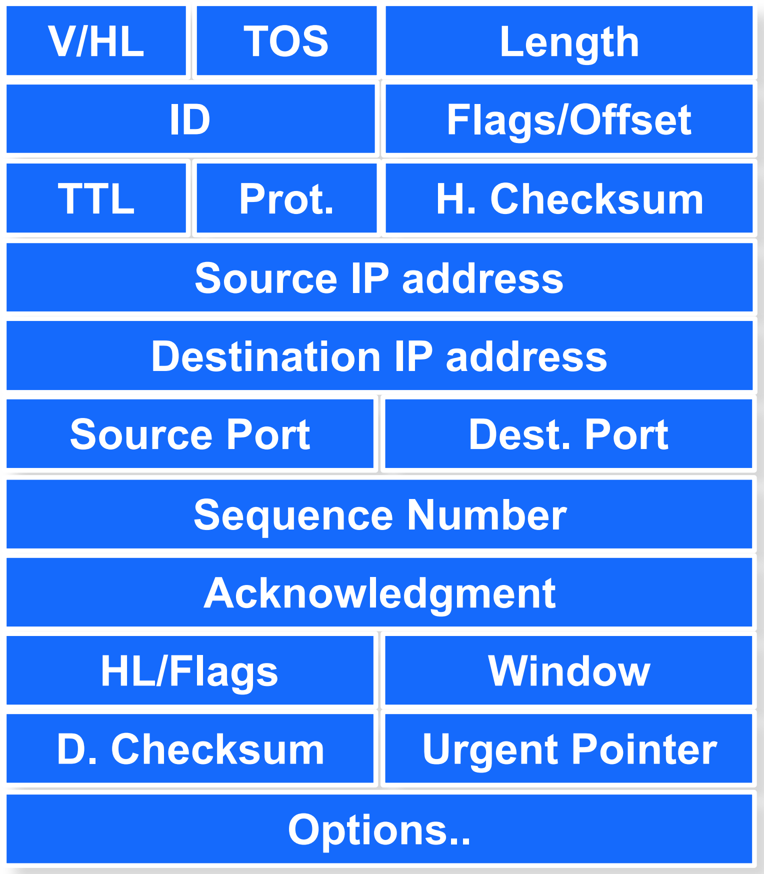


Firewalls

- The goal of the firewall is to control what traffic enters and leaves a network.
  - Creates a trust boundary: people outside of the firewall are trusted less than people inside the firewall
  - Similar to putting a guard and the door and checking ids
- Firewalls alone do not offer sufficient security.
  - Still have to be concerned about security breaches from within the organization
  - Every organization has material that require different levels of secrecy
  - But firewall limits how much traffic has to be monitored
  - Can also help with denial of service attacks (e.g. SYN flooding)

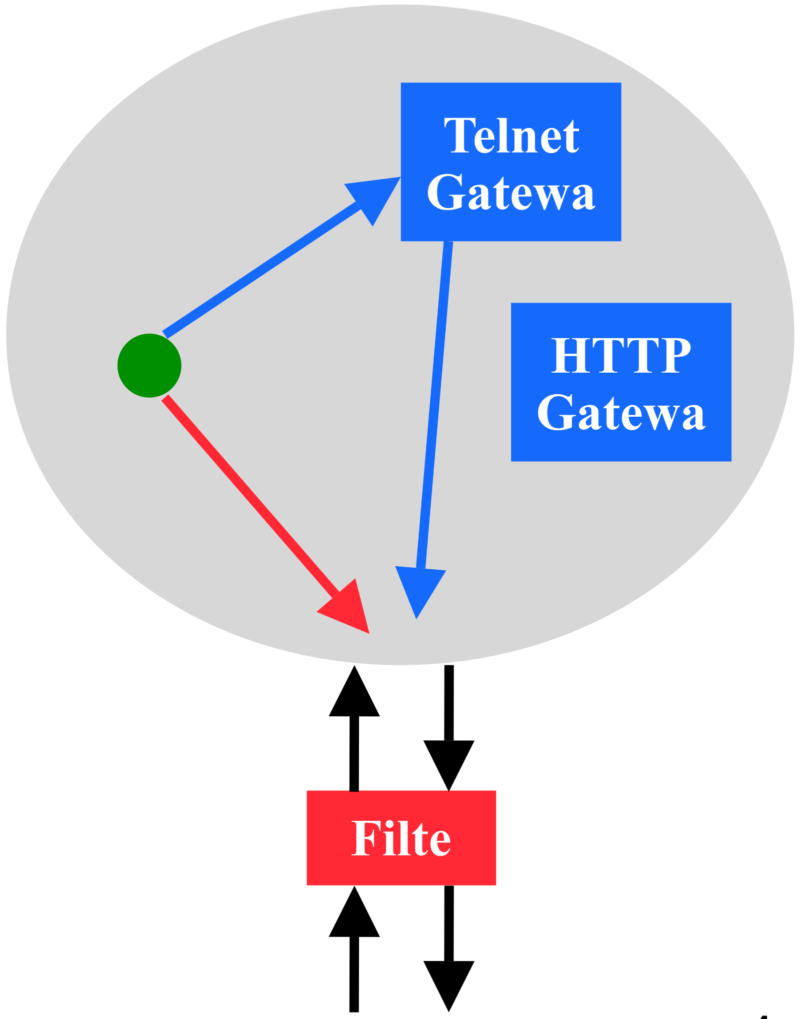
Filter-based Gateways

- A filter classifies packets based on the header.
  - IP addresses
  - port numbers
  - Protocol and message types
  - Connection information
- Filter decides what packets go through and packets are dropped.
  - No telnet, only outgoing web connections, ...



Application Gateways

- The application-level connection is terminated at the gateway and a separate connection is established over the external network.
- The gateway can monitor contents of messages since it “understands” the application.
  - Application header versus data
- Can be combined with the use of filters.
  - E.g., the filter only forwards connections from an application gateway



AAA

- Authentication, Authorization, Accounting.
  - Process used whenever users access a commercial ISP
  - ISP wants to know who you are
  - ISP will verify that you are allowed to get service
  - ISP will want to keep track of your use of the network for charging and auditing purposes
- Example protocol is RADIUS.
  - Example uses: dialup access to large access providers
  - IETF standard

Detecting Attacks: Intrusion Detection

- What to detect?
  - Intrusion attempts
  - Successful intrusions, i.e. compromised hosts
- Detecting intrusion attempts
  - Filter and log certain packets
  - Analyze the logs
  - Example: snort
    - www.snort.org

Detecting Compromised Hosts

- Certain files on a compromised host may be modified
  - » E.g. cracker installs “rootkit”
- “Integrity check”
  - » Construct a database that stores an encrypted hash of each important file
  - » Check all the files periodically (e.g. every day)
  - » Example: tripwire
    - – www.tripwire.org