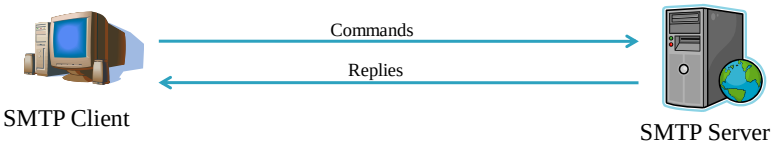
Web Security

Email security

SMTP protocol

- Simple Mail Transfer Protocol
- First defined in RFC821 (1982), later updated in RFC 2821 (2001) and most recently in RFC5321 (Oct 2008)
- Communication involves two hosts
  - SMTP Client
  - SMTP Server
- TCP on port 587 or 25 is used for communication

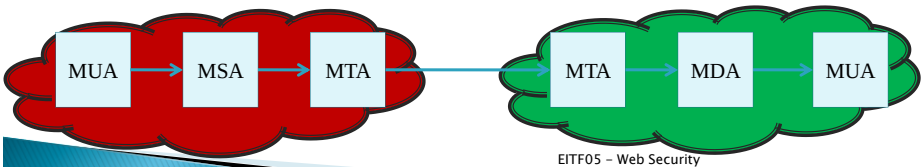


Email architecture

- Mail User Agent (MUA): email client, provides the user interface
  - Eudora, pine, outlook, kmail, ...
- Mail Submission Agent (MSA)
  - Usually implemented with MTA
- Mail Transfer Agent (MTA): The software used to transfer emails between servers
  - Implements SMTP
  - Sendmail, Microsoft Exchange Server, ...
- Message Delivery Agent (MDA): The software that delivers received email to the MUA
  - procmail
  - Usually implemented with MTA

SMTP Systems

- Originating SMTP system – Introduces mail into the Internet, or into another environment
- Delivery SMTP system – Ultimately receives the mail and delivers it to the mail user agent
- Relay SMTP system – Passes on mail from a client to another server
  - Adds trace information in header, nothing more
- Gateway SMTP system – Receives mail from one transport environment, and releases it into another environment
  - Allowed to change information in header (not received header) in order to comply with new environment
  - Can include spam filter to reduce traffic
  - SMTP may or may not be used on both sides

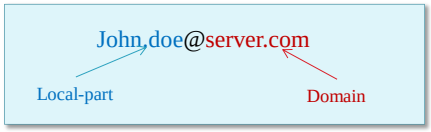


Commands

- HELO – Initiate a mail transaction
- EHLO – Extended HELO
- MAIL FROM: - Provides sender identification
- RCPT TO: - Provides recipient identification
- DATA - Provides message. End of data indicated by ”.” on an empty line
- VRFY – Ask server to verify an address
- EXPN – Ask the server to expand a mailing list
- QUIT – Close transmission channel
- RSET – Reset current transaction
- HELP – Ask server to return help of commands

Commands

- Commands and arguments are not case sensitive
  - ”VRFY”, ”vrfy”, or ”VrFy” can be used
  - However local-part of mailbox can be case sensitive



  - John.doe@server.com may not be same as john.doe@server.com.
  - john.doe@server.com is always same as john.doe@SERVER.com

Replies

- 4 main categories

- 2XX – Positive Completion reply
  - Action successful, you can start another one
- 3XX – Positive Intermediate reply
  - Command accepted, but you are not done yet
- 4XX – Transient Negative Completion reply
  - Command not accepted, but you are encouraged to try it again. It might work next time
- 5XX – Permanent Negative Completion reply
  - Command not accepted, do not try the same thing again

Some security consideration

- Possible to disable VRFY
  - Can be used by spammers to check for valid email addresses
- Possible to disable EXPN
  - Can be used to harvest addresses from mailing lists

Example, send a message

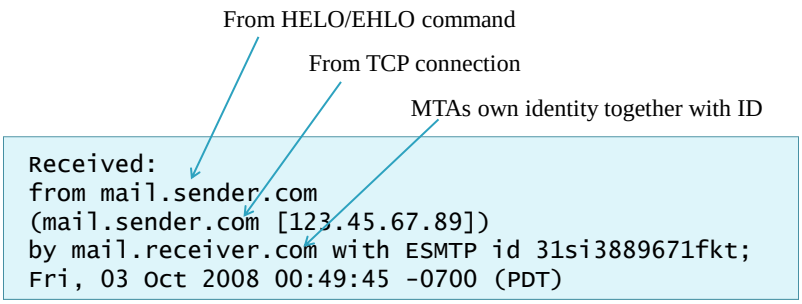
S: 220 server.com Ready
C: EHLO client.com
S: 250-server.com greets client.com
S: 250-8BITMIME
S: 250-SIZE
S: 250-DSN
S: 250 HELP
} Extensions supported by server
C: MAIL FROM:<sender@client.com>
S: 250 OK
C: RCPT TO:<rec1@server.com>
S: 250 OK
C: RCPT TO:<rec2@server.com>
S: 250 OK
C: DATA
S: 354 Start mail input; end with <CRLF>.<CRLF>
C: This is my message
C: .
S: 250 OK
C: QUIT
S: 221 server.com Service closing transmission channel

Mail Headers

- Included in DATA part
- Usually hidden by MUA
- From: and To: header is provided by sender
  - Can easily be forged
- Return-path is added by last SMTP server
  - Used for e.g., error messages if mailbox does not exist
  - Derived from the MAIL FROM command
  - Can have several other names (bounce address, return path, envelope from etc)
- Received: Information added by each involved SMTP server
- Message-id: An ID for the message which is added by the first SMTP server
- X-Header: Headers starting with X- are not part of the standard but used for information only
  - X-Mailer identifies the mailing program
  - X-Headers can be added by anti spam software, anti virus software etc

Received Headers Added by Servers

- An MTA must add the header ”received” when it receives/forwards a message
- Example



Sending a forged email

- It is easy for anyone to connect to port 25 of a mail server and send an email
  - Commands can be chosen arbitrarily
- Without additional checks of involved parties emails can easily be forged
- Headers can be used to track email and (hopefully) find who initiated the email

Example of forged email

```
Return-Path: [fake@anywhere.com]
Received: from smtp.server1.com (smtp.server1.com
[134.72.98.54]) by smtp.server2.com with ESMTP id
73659812; Fri, 12 Dec 2007 13:46:54 -0400 (EDT)
Received: from google.com (dklku64.someISP.com
[234.56.67.78]) by smtp.server1.com; Fri, 12 Dec 2007
10:45:28 -0700 (PDT)
Date: Fri, 12 Dec 2007 10:45:28 -0700 (PDT)
From: cheap products <cheap@gmail.com>
To: something@somewhere.org
Subject: The best offer only for you
```

- Here we can see that the bottom received header stems from a forged email
  - Claims that google.com was the SMTP client while it was in fact someone else (234.56.67.78)
- IP used in TCP connection cannot be spoofed

MX-records

- DNS entry specifying where to send email
  - Final delivery server
  - Relay
  - Gateway

1. MTA makes DNS query for MX record for recipient's domain name
2. Receives list of servers that can receive the mail
3. MTA tries to establish SMTP connection to MTA given in MX record
4. Each server on the list has a priority number
   - Lower number → higher priority

MX lookup for gmail.com

```
C:\>nslookup -type=MX gmail.com
Server:  ***
Address:  ***

Non-authoritative answer:
gmail.com       MX preference =  5, mail exchanger =      gmail-smtp-in.l.google.com
gmail.com       MX preference = 10, mail exchanger = alt1.gmail-smtp-in.l.google.com
gmail.com       MX preference = 20, mail exchanger = alt2.gmail-smtp-in.l.google.com
gmail.com       MX preference = 30, mail exchanger = alt3.gmail-smtp-in.l.google.com
gmail.com       MX preference = 40, mail exchanger = alt4.gmail-smtp-in.l.google.com

     gmail-smtp-in.l.google.com internet address = 74.125.43.27
alt1.gmail-smtp-in.l.google.com internet address = 72.14.213.27
alt2.gmail-smtp-in.l.google.com internet address = 74.125.157.27
alt3.gmail-smtp-in.l.google.com internet address = 74.125.47.27
alt4.gmail-smtp-in.l.google.com internet address = 74.125.91.27
```


Server priority in MX-records

- Priority can help with load balancing

> If there are multiple destinations with the same preference and there is no clear reason to favor one (e.g., by recognition of an easily reached address), then the sender-SMTP MUST randomize them to spread the load across multiple mail exchangers for a specific organization.
>
> *RFC5321*

- Use lower priority servers as backup servers
  - If server is offline:
    - No backup server: MTA will queue message and retry sending later. MTA has no idea when server is online again
    - With backup server: MTA sends message to backup server. Backup server will (potentially) know when primary server is online again → more efficient delivery
  - Backup servers sometimes have worse spam filtering than primary servers

MX lookup for hotmail.com

```
C:\>nslookup -type=MX hotmail.com
Server:  ***
Address:  ***

Non-authoritative answer:
hotmail.com     MX preference = 5, mail exchanger = mx2.hotmail.com
hotmail.com     MX preference = 5, mail exchanger = mx3.hotmail.com
hotmail.com     MX preference = 5, mail exchanger = mx4.hotmail.com
hotmail.com     MX preference = 5, mail exchanger = mx1.hotmail.com

mx1.hotmail.com internet address = 65.55.92.168
mx1.hotmail.com internet address = 65.55.92.184
mx1.hotmail.com internet address = 65.54.188.72
mx1.hotmail.com internet address = 65.54.188.94
mx1.hotmail.com internet address = 65.54.188.110
mx1.hotmail.com internet address = 65.54.188.126
mx1.hotmail.com internet address = 65.55.37.72
mx1.hotmail.com internet address = 65.55.37.88
mx1.hotmail.com internet address = 65.55.37.104
mx1.hotmail.com internet address = 65.55.37.120
mx1.hotmail.com internet address = 65.55.92.136
mx1.hotmail.com internet address = 65.55.92.152
```


Open Mail Relays

- Anyone can connect to the SMTP server and send messages from anyone, to anyone
- Will provide anonymity of real sender
- Typically used by spammers
- Used to be very common – less common now

- SMTP servers that are not open relays
  - Will deliver messages to its supported domain
  - Will send messages from IP addresses it supports

DKIM

- DomainKeys Identified Mail, described in RFC4871
- Digital signature of message put in message header
  - Certificates are not used
- Associates a domain name to an email message
- Verification that domain has not been spoofed
  - Assuming receiver knows that DKIM should be used for that domain
- Additionally provides integrity protection of message
- Hash algorithm: SHA-256, (SHA-1)
- Signature Algorithm: RSA
- Hashes, signatures and keys represented in base64

DKIM example

```
DKIM-Signature:
v=1;                Version
a=rsa-sha256;       Algorithms used
c=simple/relaxed;   Canonicalization (How message was prepared)
d=gmail.com;        Domain
s=gamma;            Selector
h=domainkey-signature:received:received:  Signed headers
message-id:date:from:to:subject:mime-
version:content-type;
bh=9gicsZnlcLK7yYh6VIrgyAMMRZiWsSbWqSPIhc Hash of body
78RRk=;
b=k4ofvpHPkaQmvuSoGVhRrnCsPK+JEuv9KUrZO7a Signature
iypvf/6Y1N2iIatvLvdzwOnZX/W6Kxyx6Z4Ybuk8D
qk/vNTIE7Jpy+GQUUHFvM0NFtmZo1CbGRvo8DdHnX
RBB/qWwlV+Z6wxw/mq7lNuJknVprOAaTLws5mwcZ+
AWL8KwHgO=
```

DKIM example

- SMTP server can check the signature by receiving the public key through DNS
- Always stored in subdomain ”_domainkey”
  - Selector is subdomain of _domainkey
- Example:

```
>nslookup -type=txt gamma._domainkey.gmail.com
Server:      ***
Address:     ***

Non-authoritative answer:
gamma._domainkey.gmail.com      text = "k=rsa; t=y;
p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDIhyR3oItOy22ZOaBrIVe9m
/iME3RqOJeasANSpg2YTHTYV+Xtp4xwf5gTjCmHQEMOs0qYu0FYiNQPQogJ2t0M
fx9zNu06rfRBDjiIU9tpx2T+NGlWZ8qhbiLo5By8apJavLyqTLavyPSrvsx0B3Y
zC63T4Age2CDqZYA+OwSMWQIDAQAB"
```


SPF records

- Sender Policy Framework (originally Sender Permitted From)
- Background: Many spammers send emails from fake domains
  - Find an open relay somewhere, send message from domain server.com

```
MAIL FROM: john.doe@server.com
```

- Idea: Let server.com decide which mail servers that can send mail from server.com
- A domain can put an entry in the DNS that lists the SMTP servers that are allowed to send mail from the domain
- Server MTA receiving mail can check DNS record for servers domain
  - If originating client MTA is not allowed to send from given domain, then server MTA can choose to not accept message
- Can be seen as a backwards MX record
  - MX record tells us where to send a mail going to a specific domain
  - SPF tells us from where we are allowed to send mail from specific domain

SPF records

- SPF (and/or TXT) record specifies rules
  - `all`: Any IP matches
  - `a`: IP match if domain has A record that resolves to sender's IP
  - `mx`: IP match if domain has MX record that resolves to sender's IP
  - `ipv4`: IP match if within given interval
- and qualifiers
  - `+`: pass
  - `?`: neutral (default)
  - `~`: suspicious
  - `-`: fail
- Examples: `v=spf1 a mx -all`

  `v=spf1 a:sub.server.com ipv4:1.2.3.4/24`

DMARC

- Domain-based Message Authentication, Reporting and Conformance
- There are issues with both DKIM and SPF:
  - What are the effects?
  - How does client handle bad signatures or IP's?
  - DKIM: Sender does not know if there are emails with bad signatures
  - SPF: Mistake(s) in list of allowed IP addresses?
- DMARC combines DKIM and SPF with application rules and a feedback system
- Three components:
  - DKIM
  - SPF
  - Alignment (From header verified against DKIM and SPF domains)

DMARC

- DMARC record uses ”tag=value” syntax
  - v: Version, must be DMARC1
  - p: Policy, one of *none*, *quarantine* and *reject*
  - pct: Percentage (0-100) of messages to which the policy is applied (default 100)
  - rua: URI to send aggregate feedback to
  - ruf: URI to send forensic feedback to
  - adkim: Alignment mode for DKIM, strict or relaxed (default)
  - aspf: Alignment mode for SPF, strict or relaxed (default)
  - sp: Same as p–tag but applied to subdomains
  - ri: Report interval for aggregate reports (seconds, default 86400)
  - rf: Format for forensic reports
- _dmarc.amazon.com

```
v=DMARC1;
p=quarantine;
pct=100;
rua=mailto:dmarc-reports@bounces.amazon.com;
ruf=mailto:dmarc-reports@bounces.amazon.com;
```

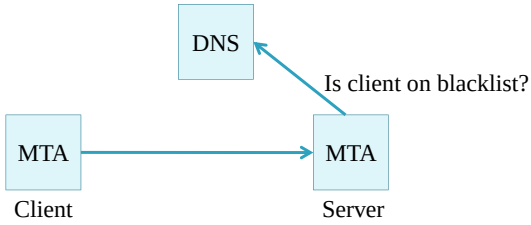

Anti-spam

- About 100 billion spam emails sent per day

- Several methods used to combat the problem with spam
- Can be divided into filtering and blocking
- We will look at
  - DNS Blacklists (DNSBL)
  - Greylisting and nolisting
  - Hashcash
  - Statistical Filters
  - Hybrids

DNSBL

- DNS blacklists, also called DNS blocklist
  - Also known as RBL (Realtime Blacklist)
- A list with IP addresses that are (considered by blacklist provider) used for spamming
- IP can be identified by e.g., honeypots
- Server MTA can check if client MTA is on list



- If client is reported in blacklist, server can take appropriate action
- Can be used as one part in a spam scoring system

DNSBL

- Easy to use – IPs can be checked using DNS queries

```
IP(r).dnsbl.server.com
```

IP(r) is the reverse byte ordering of the IP address to check

- Some big ones: spamhaus, spamcop
- Some are more aggressive than other
- Problem: False positives and false negatives
- Problem 2: What is a positive and what is a negative

DNSBL

- Some IPs can be blocked by one blacklist but not with another
- Example: Check IP 209.237.225.253

```
>nslookup 253.225.237.209.zen.spamhaus.org
Server:      ***
Address:     ***

**Server can't find 253.225.237.209.zen.spamhaus.org: NXDOMAIN

>nslookup 253.225.237.209.spam.dnsbl.sorbs.net
Server:      ***
Address:     ***

Non-authoritative answer:
Name: 253.225.237.209.spam.dnsbl.sorbs.net
Address: 127.0.0.6
```


DNSBL

- Advantages:
  - You can choose the blacklist that suits you
    - Aggressive lists will prevent most spam but also reject some legitimate emails
    - Conservative lists will miss some spam but are less likely to reject legitimate email
  - Message can be rejected before it is actually sent since server can drop connection with client immediately
- Drawbacks:
  - Same mailserver can be used by both spammers and legitimate users
  - If a mistake puts you on a blacklist, it may be difficult to get off the list

URI DNSBL

- URI version of DNSBL
- Targets URLs and IPs in message body
- Server MTA can check message body for links that are not likely used in normal email, but known to be used in spam
- Works in the same way as DNSBL
  - URI is checked with DNS

Greylisting

- Background: Many spam programs do not fully comply with SMTP
  - They might not retry sending an email that was previously rejected
- Idea: Always reject unrecognized transactions
- For incoming messages, look at

  (SMTP Client IP, sender address, receiver address)
- If not previously used, then save in database and temporarily reject message (Transient Negative Completion reply)
- If recently used, then accept message

- Advantages:
  - Easy and not very resource consuming compared to some other methods
  - Can be used before other spam filters to reduce their workload
- Drawbacks:
  - Email is no longer ”realtime”
  - Relies on legitimate servers implementing the retry

Nolisting

- Background: Many spam programs do not fully comply with SMTP
  - Some only try the highest priority server in the MX record
  - A few try the lowest priority server in the MX record only, assuming it is used as backup and does not have good spam filters
- Idea: Let the server with highest priority be non-existing

```
server.com:
       10 dummy.server.com
       20 real1.server.com
       20 real2.server.com
       30 real3.server.com
```


Hashcash

- Pay to send an email, but receive for free
- Background: Computers and connections are fast, sending many emails is fast and cheap
- Idea:
  - If it takes 1 microsecond to prepare a message to send, then many messages can be sent in a short time
  - If it takes 1 second to prepare a message, not so many can be sent in reasonable time
- Idea similar to
  - Key strengthening
  - Slow hash functions used to strengthen password protection
- Hashcash is asymmetric in the sense that it is expensive to prepare the message but cheap to verify that it was prepared with hashcash

Hashcash

- The string

```
*ver*:*bits*:*date*:*resource*:[*ext*]:*rand*:*counter*
```

is hashed with SHA-1
  - *ver* is version number (currently 1)
  - *bits* indicates how costly the function is for sender
  - *date* gives current date
  - *resource* is recipients email address
  - *ext* is extensions
  - *rand* is a random number (separates different senders)
  - *counter* is a counter value
- If the first *bits* bits of hash are zero, then string is added in message header
- Otherwise, increase counter by 1 and hash again

```
X-Hashcash: 1:20:131015:receiver@someserver.com::7239672987:35976
```

- String verified by receiver and saved in database
  - Can only be used once

Hashcash

- Advantages
  - Will not block mail from legitimate senders
  - Not costly for receiver
- Drawbacks
  - Botnets can be used instead of single computer preparing messages
  - Slow computers have more problems than fast ones
    - Embedded devices

Statistical filtering

- Background: Some words are more common than others in spam emails
- Idea: Construct an algorithm that can sort out spams based on the content of the message
- Bayesian filtering
- Use training data to teach an algorithm how to separate spam from legitimate emails
- Let D be the event that a document contains a set of words w_0, w_1, w_2, ...
- Let S be the event that the document is spam and S' the event that it is not spam
- Then

$$p(D|S) = \prod_i p(w_i|S) \qquad p(D|S') = \prod_i p(w_i|S')$$

- We assume that words occur independently

Statistical filtering

- Use Bayes law

$$p(S|D) = \frac{p(S)p(D|S)}{p(D)}$$

- Thus

$$p(S|D) = \frac{p(S)}{p(D)} \prod_i p(w_i|S) \qquad p(S'|D) = \frac{p(S')}{p(D)} \prod_i p(w_i|S')$$

- Divide:

$$\frac{p(S|D)}{p(S'|D)} = \frac{p(S)}{p(S')} \prod_i \frac{p(w_i|S)}{p(w_i|S')}$$

- Take logarithm

From training

$$\ln \frac{p(S|D)}{p(S'|D)} = \ln \frac{p(S)}{p(S')} + \sum_i \ln \frac{p(w_i|S)}{p(w_i|S')}$$

- Spam if log-likelihood ratio is larger than some threshold, e.g., 0

Hybrid filters

- Some or all of the previous methods can be combined
- Then passing/failing a test can contribute to a total ”score” for an email
- If score is higher than threshold then email is considered as spam
- An open source implementation is SpamAssassin
- Can be used in MTA, MDA and/or in MUA