COMP 535 ASSIGNMENT #2

SIMON FOUCHER (260223197)
AMIN MIRZAEE (260209556)
IVAN ASLAMOV (260228181)

PART 1: DNS

1. DNS can map a single name to multiple IP addresses as shown in one configuration. What is the cost (cost is measured in required DNS server capacity) of using a single name for multiple IP addresses?
The cost in performance will be better than mapping a name onto another one, because a single host request will suffice to resolve the IPs of the desired domain. Typically, the server will return the list of IPs associated with the domain, changing their order from one request to the other. The host typically chooses the 1st address it receives, so this takes care of load balancing. From this point, the host can access the domain directly by reachin out to one of the provided IPs. The only extra cost for the DNS server is a bit more memory to store the extra addresses in the RR.

2. Several examples of DNS mapping multiple names to a single IP address were included in the above experiments. What is the cost of using multiple names? Compare this cost with the one for mapping a single name to multiple IP addresses. As designed, which mapping is more efficiently supported by the DNS infrastructure?
It is less efficient to map a name onto another name. If a domain is resolved to an IP address immediately, the host can then establish routing communication to the remote domain. In contrast, if the original query returned a list of NS, the host will need to contact these NS and dig down their hierarchy sequentially until it finds one that can provide an IP. At minimum, this doubles the required processing time for a single request by the DNS infrastructure.

3. Fault tolerance is an important consideration in DNS. How is it implemented? Before you answer this question, use dig to query for the name servers for large number of academic institutions in US and Canada. Do you see a pattern in how the name servers are configured for each domain?
Fault tolerance can be implemented by having the authoritative NS of a domain answer queries with the name of many name servers who are aware of the requested domain, instead of returning the IP of the requested domain. The host can then send a second query to any of those name servers in order to resolve the IP of the desired domain. If one of the servers happens to go down, the host will still be able to access the domain by sending the query to the other Name Servers. This redundancy can be implemented at all levels of the DNS hierarchy to provide DNS fault tolerance at many levels. Large institutions thus have their name configured on many redundant nameservers.

4. Does this approach to provide fault tolerance (identified above) provide attack tolerance as well? What type of attacks can you tolerate? Do you see any cases where the fault tolerance technique can make DNS more vulnerable?

This scheme provides some sort of protection against generalized DoS attacks. Since the DNS load is distributed over many machines, the attacker will need to be able to swing a lot more bandwidth in order to take down a domain. This can make DNS more vulnerable if one of the NS has specific access vulnerabilities. If an attacker can introduce himself as root into one of the NS that are responsible for routing traffic to a domain's IP, he could alter the routing table and change the domain's IP to a phishing site with a very long TTL. The incorrect results returned can then be cached for extremely long periods of time, causing extreme disturbances in service. The other non-

compromised servers will eventually stop being queried due to the caching, resulting in a complete takeover of that name.

PART 2: SMTP

1. Give a brief security analysis of the mail setup described in this experiment. Can the current setup be easily exploited to send spam or other unsolicited e-mails?

Yes. It would be easy to modify the 'myorigin' entry with a spoofed origin for the spam messages. Afterwards, when messages are sent, the recipient would not be able to track down the sender just by looking at the mail header.

2. In your opinion, is email spam a result of deficiencies in email delivery infrastructure, administration of a sound infrastructure, malicious users who cannot be controlled by any means, or a combination of the above. Provide valid reasons (from observations from doing this experiment) to support your position on the email spam problem.

The blame could be put on the email delivery infrastructure, but email traffic is so dense that it needs to be highly automated in order to work. The blame therefore has to be put on malicious people. No matter how many filters we put in place to avoid spam, if a human can design a filtering algorithm, chances are a human can break it.

Another argument shifting the blame to malicious users is the fact that this problem is also observed in many other fields of activity, even regular mail. If it were email specific flaws, this problem would be limited to email only. Since we can observe similar conflicts in many other sectors of activities, we cannot shift the blame to the specific systems/algorithms in place.

Whenever these issues are observed, the scenario is typically the same. On one hand, we have an institution designing policies to handle quick processing of massive legitimate traffic. On the other hand, we have malicious users that try to find a flaw in the system and exploit it.

3. Reputation management is used in e-commerce sites to rank sellers and/or commodities sold online. Buyers take the reputation measures into consideration when they make the buying the decisions. Explain how a reputation management scheme can be used in e-mail. What are the major hurdles in deploying a reputation management system to cut-down on spam? (gmail is already using some form of reputation management for spam control.)

Like gmail, have a 'report spam' function on the mail server. Whenever it is pressed, the originating mail server gets -1 point. When a mail server gets too many negative points, it can be flagged as a source of spam. The observed reputability of each server should then be reported to a number of spammer databases, to better keep track of which addresses send out spam. Automatic content analysis would also be necessary as an added level of protection. Additionally, mail servers spoofing the originating email address (can be determined through simple DNS lookup) can be blocked. Deploying measures such as these is a difficult task, however, because not every email system will adhere to these guidelines. Also, the task of setting up a new mail server becomes very difficult, as initially all mail originating at this server will be sent to spam.

The distributed nature of email makes it very difficult to adopt a good strategy for tackling the spam problem. As cloud computing becomes more mainstream, however, email as it is today will become a thing of the past. Google Wave is a good example of what internet communication is transitioning to. In a system such as this, where messages are all stored on a central cloud, users/IP addresses marked as spammers could be required to pass some sort of CAPTCHA test. While spam may still get through this way, it will be impossible to send the volumes of it necessary to achieve any worthwhile return.