

February 13, 2013

**Question 1** *IP Spoofing*

(15 min)

You are the network administrator for a large company.

- (a) Your company will be held liable for any spoofing attacks that originate from within your network (i.e., packets leaving your network with spoofed IP header information). What can you do to prevent spoofing attacks by your own employees?

You now want to evaluate the risk your employees face from spoofed IP packets originating from outside the network.

- (b) Assess the likelihood and dangers of spoofed IP packets that use TCP as the transport layer protocol. What applications might be vulnerable to such an attack? How does this change with UDP?
- (c) What can be done to prevent parties outside your network from sending your employees spoofed traffic that impersonates your own employees.
- (d) (*Optional*) Now consider that your network has multiple links to the internet. Is there anything you can do to reduce the possibility of outsiders successfully sending your employees spoofed packets?

**Solution:**

- (a) Inspect the source IP address of all outgoing packets. If a packet has an address from outside the range assigned to your network, block the packet. This is called *egress filtering*.
- (b) Recall that TCP uses a 3-way handshake to establish a connection. As part of that handshake each side must agree upon a pair of valid sequence numbers. In order to successfully spoof a new connection, or to inject a spoofed packet into an existing connection, the attacker must either know or correctly guess the valid sequence numbers.

For a *blind* spoofing attack, without the use of any other attack techniques, the likelihood of correctly guessing the sequence numbers is quite small. (You'll work out this math in a later assignment.)

However, if the attacker is on-path (can eavesdrop) then spoofing TCP connections is quite easy.

An example of an application that would be vulnerable to such an attack would

be HTTP. Later in the semester we will discuss TLS (SSL) and how that could help mitigate this attack.

Contrasting to TCP, UDP does not require a 3-way handshake. The attacker must only know the right source port from which the victim has started an outgoing connection. Thus, spoofing UDP packets requires much less effort. If application layer protocols wish to defend against this kind of attack they must develop their own defenses.

An example of an application that would be vulnerable to such an attack is DNS. We will explore DNS attacks in more depth in the next question.

- (c) Packets originating from outside your network should never have a source IP address from inside your network. Using this fact you can filter (block) incoming packets that contains source IP that belongs to your own network. This is called *ingress filtering*.
- (d) It is highly dependent on how your system is setup. If you know what IP addresses are associated with the networks behind the different links, and a packet comes in on a link that does not match with the IP addresses associated with that network, you can filter out those packets. This is again *ingress filtering*. It is not always possible to associate a set of IP addresses with a link connection. ISPs can do this for traffic coming from their edge customers (on separate autonomous systems).

## Question 2 *DNS*

(15 min)

Recall that in a *blind* DNS spoofing attack, the attacker tries to guess the identification number of the DNS request sent by the victim.

- (a) For the following, assume that the victim's DNS resolver cache does not contain a record for the domain the attacker is targeting:
  - i. In a blind spoofing attack, the attacker must know when a DNS request is about to be made by the victim so that the attacker can respond with their attack responses. Recall from lecture how an attacker might learn this information.
  - ii. What can an attacker do if they successfully gets a victim to believe their bogus DNS mapping?
  - iii. How can an attacker avoid having to carry out this attack for every time the victim visits the targeted domain?
- (b) Now assume that the victim's DNS cache has a genuine NS record for the domain the attacker is targeting.
  - i. Can the attacker still be successful at poisoning the A records for some of the names belonging to the domain?
  - ii. Can the attacker poison the NS record of this domain? If yes, how?

### Solution:

- (a)
  - i. Lure a victim to your web site, which contains an image (or many images) pointing to the site whose DNS record you would like to attack (e.g., `google.com`). When you see that a victim has contacted your site, you know they are about to make a DNS request for `google.com`, so you initiate the attack at this point.
  - ii. The attacker is in control of the content of any request made to a host-name whose DNS record has been successfully spoofed. For example, if the attacker managed to get the victim to accept a bogus DNS record for `google.com`, then any subsequent request to `google.com` will actually go to a domain of the attacker's choosing. The attacker might get you to reveal your password to the fake `google.com` at that point. DNSSEC and TLS (SSL) are useful techniques to mitigate the impact of this attack. We will learn about both these topics later in the semester.
  - iii. DNS records are cached, so the attacker might set a long TTL (time-to-live) so that the bogus DNS record will live in the cache for a long time. The attack will succeed for as long as the bogus DNS record lives in the cache.
- (b)
  - i. The attacker can still poison A records of subdomains. For example, if the victim has an NS record for `foo.com` in the cache that points to `ns.foo.com`, then the attacker can simply force the client to lookup random subdomains such as `qt0378hr.foo.com`. The victim only knows who to ask, namely `ns.foo.com`, but does not know the answer for that query.
  - ii. The attacker can also poison the NS record using the Kaminsky attack[1]. In this case, the attacker can specify the same NS record in the *authority section* of the reply to tell the victim that `ns.foo.com` is responsible for the domain `foo.com`. Moreover, the attacker also provides an A record for `ns.foo.com` in the *additional section* pointing to a bogus address. This extra information is also known as the *glue record*.

## References

- [1] Reliable DNS Forgery in 2008: Kaminsky's Discovery.  
<http://www.cs.berkeley.edu/~daw/teaching/cs261-f09/reading/matasano-kaminsky-dns-forgery.html>.