hw07

April 2017

1 VPNs

1. A VPN is a virtual private network that can be used to connect to a private network across a shared, public network. In terms of this article, let's say you want to work remotely - you don't necessarily want to access company material on a public network since this leaves communications susceptible to attacks, such as MitM attacks. Instead, you'd use a VPN to connect to your company's intranet, and once you're on this private network work securely.

   Concretely, let's say you're using OpenSSH, a type of VPN. This'll establish a point-to-point connection from your client to some private network. Furthermore, it'll make sure your communications to the private network are encrypted. Thus, you'll have access to all the resources on your company's intranet via encrypted transmissions.

2. As the paper suggests, using a VPN to connect to some company's intranet, and then once connected, allowing a user to work as if they were no longer accessing resources remotely relies on a 'perimeter security model': if you can gain access to the company's intranet, you're assumed to be a trusted user, while anyone outside the walls is deemed dangerous. The paper contends that this is an outdated model.

   Instead, BeyondCorp doesn't differentiate between some trusted company intranet vs. dangerous public internet. There's no need to tunnel into your company's intranet. Instead, corporate applications are on the internet, with all access to company resources regulated through a combination of device and user credentials, which specifies who gets access to what. Only "managed devices", managed through certificates, are allowed access to the company's network. Once you connect to the company's network, users are identified in a User database and given short-lived tokens. With these two levels of authentication, the company can, with more granularity, restrict who gets access to what; a valid device certificate is necessary to access the company's network, but after that the user's credentials specify what he or she can do on that network (and these can be easily changed depending on job role, promotion, etc).

3. I think what happens when someone tries to access a google application via some unsecured WIFI, for example while using WIFI at a coffee shop, is a public domain name for that application points to an access proxy through which the device's credentials and, later, the user's credentials are checked. If the device is a valid managed device, access is granted and then the user's permissions are specified via their user credentials. What's key, here, is that the user isn't using a VPN to remotely connect to their company's intranet. Instead, everything is on the internet, access is simply regulated with the device's and the user's credentials. No longer is there this idea of a perimeter beyond which users are considered trusted. To simply gain access to the network depends on whether you have a trusted device, and then beyond that your individual credentials are what determine your level of access.

4. I think the big ways a VPN provides protection, even from an ISP, is through encryption and masking of the final location of your requests. So, if you're using a VPN like OpenSSH, since messages between a client and server are being sent in an encrypted way, it's not possible for an ISP to determine the content of the payload of the messages.

   Furthermore, if you're accessing a website via a VPN, your ISP will realize you're sending packets to the address of the VPN, but it won't know what website you're visiting through the VPN. Thus, the actual websites you're visiting will be masked and inaccessible to your ISP.

2 Arp Example

Let's say Host A tries to send a packet to IP address: 128.148.32.12.

1. First thing to notice is that the destination of the packet is outside Host A's subnet, therefore Host A is going to have to make use of its default gateway to communicate with other networks outside of its local area.

2. In order to locate the router at IP 192.168.1.1, and because Host A's ARP cache is empty, A will send out broadcast address: FF:FF:FF:FF:FF:FF, which should return the MAC address of the router in Host A's subnet: 11:11:11:11:11

3. Host A will send its packet to the router at MAC address 11:11:11:11:11. The router, whose ARP cache is also empty, will try to identify a subnet to forward packet 128.148.32.12 to by sending out broadcast address FF:FF:FF:FF:FF:FF. Since none of the subnets connected to the router are of the form 128.148.32.*, it'll forward the packet along to the internet IP: 138:16:118:162.

4. The packet will travel among routers until one finally knows the correct network to forward it to. In the case above, 128.148.32.12 refers to 'cs.brown.edu'.

3 Spoofing

1. What Host B could do is spoof Host C's IP address. Since Host A's ARP table is empty, when Host A wants to send a packet to C at IP address 192.168.1.4 it'll broadcast an ARP request to FF:FF:FF:FF:FF. IF B spoofs C's IP address, when B get's this broadcasted ARP request it'll respond with its own MAC address, which could then be put in A's ARP table. A will now associated B's MAC address with C's IP address and send future packets intended for C to B.

2. I learned in office hours that members of a subnet might broadcast an ARP request not only to resolve an IP address to a MAC address, like above, but let's say a new computer 'Z' enters a network; when it does this, no other hosts on the netork know its IP or MAC addresses. Instead of waiting for each host to broadcast a request, 'Z' might broadcast its information so that everyone can update their ARP tables.

   Since Host B only wants to affect Host A without damaging the connections of the other hosts on the subnet, it might be able to make use of this gratuitous ARP request, not broadcasted, but sent only to Host A. This directed request, only to A, will cause A to update its ARP table, replacing C's information with B's.

3. What I originally thought possible was for a single IP address to map to multiple MAC addresses. If this was possible, when host A tries to resolve the IP address for host C, it'll find multiple hosts and end up sending transmissions to B and C. However, I don't think this is possible.

   Instead, what I think B can do is:

   - First, through gratuitous ARP requests to A spoof the IP address of C such that C's IP address resolves to B's MAC address in A's ARP table.
   - Now, host B will receive transmissions from A intended for C. B, however, can examine the packets from A, specifically the **sender address** and the **target address**. If packets from A, intercepted by B, have C as their target, B can forward them along to C.
   - I'm not sure if this is possible, but if B could also edit the **sender's address** to A's address before forwarding the packets along, C will know to reply to A; A and C won't have any idea packets are being intercepted by B.

4. Similar to above, B can:

   - Through gratuitous ARP requests to A spoof the IP address of C such that C's IP address resolves to B's MAC address in A's ARP table.
   - Once B has inserted itself into A's ARP table, it can do the same thing to C, replacing A's entry in C's ARP table with B's, resolving A's IP address to B's MAC address.
   - Now, host B will receive transmissions from A intended for C and transmissions from C intended for A.
   - Using a similar technique to above, B can examine the ARP request metadata (**sender address** & **target address**) of the packets coming from A and C and forward them along to their correct destinations.
   - Thus, B will act as a way point between A and C, forwarding packets back and forth between the two as they communicate.

5. I think the key to this is realizing that IP address 128.148.32.12 is outside A's subnet and requests should therefore be sent to the router. If this is the case, B should try to spoof the IP address of the router 192.168.1.1. When A tries to send packets outside of its subnet, it'll try and send it to the router, but if B has replaced this IP address in A's ARP table with its own MAC address, B should be able to intercept packets A tries to send outside of the local network.

4 DDoS

Usually when there is a DDoS attack there's a sudden, high number of requests coming in a short period of time. A couple ways to potentially protect against this:

- limit the number of expensive requests that can be initiated by IP addresses. So, the key here is to only allow IP addresses to make a certain number of costly requests within a particular time frame. This operates under the assumption a normal user wouldn't make more requests than this. Beyond this threshold, discard additional requests.

- keep track of the kinds of requests coming in to the server. So above, you were measuring the frequency of requests. Now, look at the pattern of requests coming from IPs. Even if a computer isn't making costly requests, if it, for example, makes 100s of the same request, this might flag it as a potential bot: i.e. this doesn't seem like normal user behavior.

  – This reminds me a little of 'captchas' or the boxes you need to check to verify you're not a bot. Frequently I'll notice these if I try to log in to my account multiple times unsuccessfully. I've noticed them elsewhere, too, and I think this is a check against users initiating the same, potentially costly server request multiple times in quick succession.