hw06

April 2017

1. Onion routing:

   (a) Although using a secure Internet connection (like HTTPS) will encrypt the message body of packets sent over the Internet, this kind of communication still leaks valuable data in the form of headers; traffic analysis can determine who and where you're sending data / requests.

   This can be dangerous, if, say, you're a journalist operating in a foreign country and you're trying to relay information back home. In this instance, for a man in the middle sniffing Internet traffic, seeing data sent from your IP address to a news organization outside of the country might be enough to put you in significant jeopardy. This is why Tor is valuable.

   Tor uses onion routing. Before a message is sent, a set of nodes are chosen and arranged into a path or circuit. As the message traverses the path, layers of encryption are peeled off (like an onion), revealing where the message should be relayed to along the circuit. Thus, each node along the path only knows who sent it the message and where to forward the message, yet it doesn't know:

   - the route (i.e. other nodes along the circuit other than the node it received the packet from and the one it's relaying it to)
   - or the endpoints (i.e. if the nodes it's communicating with are the original sender or the final server)

   Thus, it's not possible for any man in the middle to decipher the origin or destination of a message, and none of the nodes have a complete mapping of the path, so they can't know either. In this way, Tor is able to mask header information that might be compromised using a normal browser and network connection.

   (b) The sender and recipient only know who they got the message from and where they're relaying the message to. They have no idea of any

other nodes along the path (since this information is still wrapped in encryption unavailable to the two nodes) or if they were the original sender or end recipient.

(c) As it says on the Tor website:

`Tor does not provide protection against end-to-end timing attacks: If your attacker can watch the traffic coming out of your computer, and also the traffic arriving at your chosen destination, he can use statistical analysis to discover that they are part of the same circuit.`

Using the same example from part (a), if I'm keeping track of requests coming from a journalist's computer and also monitoring requests on other computers, I might be able to reconstruct the path the Tor network is using, thus eliminating its effectiveness anonymizing header data.

2. SSL

(a) First, if you look at the URL, since it's a .onion site you can only access it by connecting via a Tor browser.

Let's think what a certificate does: it allows a client to be sure that a public key issued by a server is actually identified with that server.

For example, if I connect to google and want to send information securely, I'll encrypt my message with google's public key, thus protecting it from other malicious users. If an attacker was to somehow forge an SSL certificate as google, instead of using google's public key I would use the attacker's; he'd then be able to easily decrypt my message with his private key.

If a system has no built in (root) certificates there'd be absolutely no way to determine that a certificate was actually valid. You might connect to a website, with a certificate verified from a third party, but still, there'd be no way of verifying the authenticity of the third party's certificate. Regardless of how many third parties are used in the chain of trust, you essentially need to recur to some known, completely trusted base case, which is the root certificate on your computer.

3. MitM

(a) I think what's key here is you have an 'active' MitM s.t. he's not simply sniffing traffic, but might be able to alter the packets as they are sent to the server. If this is the case, the attacker might be able

to alter requests from the client indicating that the client's browser *does not* support HTTPS. This kind of alteration would prevent the server from upgrading the request and responding with HTTPS (since now it thinks the client's browser doesn't support it).

For example, from HW 3 we saw that requests in google chrome will come with a 'Upgrade-Insecure-Requests' parameter which, if flagged, will notify the server that the browser can handle a more secure version of the website than was possibly requested. A MitM could simply flag this to 0, specifying that the user's browser doesn't support a more secure version of the website, forcing the server to respond with HTTP.

4. Airport

   (a) It seems that the airport is tracking devices by their MAC address. Each device should have a unique MAC address identifying it - this would explain why changing the IP doesn't affect how the airport is tracking users and why switching to a new device (i.e. your phone), allows you to connect to the internet for 5 more minutes.

   (b) If the above is true, you could circumvent the airport's tracking by changing your MAC address every 5 minutes.

   (c) I don't think regular ISPs can track users in the same way. I think MAC addresses are primarily used in local networks, so above what **might** be happening is users at an airport are connecting to a local area switch, which is then forwarding requests to a router and then over the internet. The switch, in the above case, is what is acting as the gatekeeper.

   If you were to simply connect to the internet from home, your router would forward packets along - in this case, all that is needed is an IP address to identify the destination of return packets from the server. There's no need to pass along your device's MAC address over the internet, so ISPs shouldn't have access to it.

5. Last

   (a) Before someone boards for a flight, you could simply ask what their MAC address is and then change your computer's MAC address to theirs. Since the airport is identifying devices by this address, it'll think the user who paid for internet is still around, and thus, you'll be able to browse for free.

Alternatively, if you want to be more technical, you could sniff traffic to determine other users' MAC addresses. In office hours, I heard that you could turn on 'promiscuous mode' on your computer, which'll basically keep track of ARP requests made to your computer that would otherwise be discarded. From these broadcasted ARP requests, you could try MAC addresses you've kept track off, and hopefully, one of those would've paid for internet, which you could then piggyback off of.