**Misdirection**

**Misdirection** is where the con-artist / magician / social-engineer, someone who is trying to trick another human, gets them to pay attention to the wrong thing and calms down their interest and alertness and arousal about the right thing, so you only notice the WRONG THING.

Misdirection Philosophy

* Don’t draw attention to things unnecessarily, unless it is the deliberately wrong thing that you want people to focus on.
* Show tricks / misdirection in a way that makes it LOOK **natural**, so that your attention is NOT directed towards what it should be directed.
* You need to misdirect in a very natural way so everyone is drawn towards the misdirection and **won’t become suspicious**.

Misdirection can happen in a lot of places.

* E.g. focusing on misdirecting airport security to the passengers coming into the airport, rather than the flight crew
* **Security Theatre:** We put all our attention and energy to make sure that the one spot is right, but the little obvious, natural thing on the side where people can just walk in and out as they want because they have a small plastic ID card, that doesn’t get any attention
* When that is discovered and solved, it doesn’t matter because now we’re focusing on those two things, we’re still not looking at the other 60 things because we are bad at being attentive to everything and can only focus on a few things at once.

**Kevin Mitnick Attack**

Mitnick’s hack of Tsutomu Shimomura.

* 1994 Christmas, Shim left his computer and has gone to do some Christmas stuff
* It involved TCP / IP
  + They are two protocols for networking
  + Everyone should learn a bit of TCP / IP, because a lot of attacks are launched over the internet and they require knowing level 2/3 of TCP/IP, so you know what a TCP / IP packet looks like etc.
* IP is lowdown
* TCP is one level higher than that.
* **Good book about TCP/IP: Internetworking with TCP/IP Volume One by Douglas Comer**

**TCP**

* One of the protocols used when a computer somewhere wants to talk to a computer somewhere else, by passing messages through lots of intermediate machines along the way.
* We only care about the source of the whole thing / destination.
  + The destination is possibly a web-server, you are serving up web pages
  + The source is possibly someone who wants to read a web-page, so sends a request from his/her browser by typing in a URL that bounces around the internet in all sorts of ways and eventually ends up at the web-server, telling it that he/she wants a particular web-page.
  + Imagine it is Wikipedia that the source is requesting, so Wikipedia sends the web-page back to him/her.
  + The source’s browser gets the web-page and displays it.
  + **^This is a high-level overview of what is happening.**
  + **With networking, we go from high-level 🡪 low-level**
* Suppose you are looking at a big-page, with many many kilobytes of data on it.
  + The source sends a packet up, telling the destination what they would like 🡪 the destination sends a packet of data back with the “answer”
  + **Packets are of a certain size, they can’t get bigger than the maximum size of X kilobytes**
  + **Anything bigger than X kilobytes, the data has to be broken down into multiple chunks of data**
  + All the packets travel a particular way back and forth from the source 🡪 destination
  + Each of the packets are posted through a “postal system” which has an address of it.
  + Intermediate people get the address of the packets to look at it and then it gets passed on all the way until it reaches its destination.
  + The structure of this whole thing / these packets travelling around are doing something called IP.
  + TCP sits on top of IP.
    - **TCP is about not just sending a packet, but having a connection**.
    - Different from someone sending a letter.
    - With a letter, you have to renegotiate and decide again if you want to talk or not.
    - With a connection, once you go through all the overhead of setting it up, its very lightweight to keep communication going.
  + **TCP has the same protocol to set up an initial connection by sending three packets**
    - A source wants to connect to a web-server, so they send one packet saying:
      1. **SOURCE: S-Y-N packet** **(synchronise)**  
         “Hey, I want to initiate a connection with you”
      2. **DEST: SYN + ACK packet** **(synchronise + acknowledge)**We want the convo to go two ways, so sends both SYN and ACK)  
         “Okay I will connect with you”
      3. **SOURCE: ACK packet**

When the ACK is sent back to the source, the connection is complete

* The **SYN, SYN-ACK, ACK** = **The Handshake**
* Now both parties are ready to talk to each other.
* They will talk to each other by sending packets
* Because it’s a connection, the packets will be reassembled into the right order, rather than a random order.
* The underneath layer of things sorts all the reassembling out and makes sure that your TCP data arrives all neatly re-assembled with all the missing bits filled in and so on. And if something falls off, it has to resend it. It looks after all of that for you.
* There is now a synchronised stream of packets between the source / destination.
* The main mechanism used to sync a stream between the two parties and to keep it going is called a **Sequence Number**.
* When the **Source** starts talking to the **Server/Dest**, it sends a sequence number and vice versa.
* Then every time the source sends a message, they increase the sequence number so at the server/dest, they can reassemble all of the packets in the right order and vice versa.
* Both parties will have their own sequence number.
* The initial sequence number doesn’t matter and is random / pseudo-random etc.
* However, every subsequent sequence number increase by the number of bytes that have been sent in the last packet.
* E.g. if the source starts with a sequence number = 0, and they send 100 bytes to the destination, the next sequence number will be 100. If the next size of data they send it 600 bytes, then the next sequence number will be 700 and so on.
* At the other end, the other party can see what the sequence number is and what they should be expecting. If the sequence number doesn’t match, maybe something has gone wrong / there is some missing data and will ask the source party to resent it.
* **Sequence numbers are a way of keeping the conversation glued together and all joined up**.
* The interesting thing is that, many many people are probably connecting to the same destination and it is having lots of conversations with many sources
* **How can say, Wikipedia disentangle all those conversations? How does it know its talking to the right people and sending the right bytes to the right people etc.?**
* It does it with both sequence numbers and port numbers.
* It COULD disentangle in other ways, such as detecting characteristics of the source’s machine or the source’s typing and re-authenticate it at every point and check that its still Andy its talking to.
* It COULD check low-level things in the TCP stack and see what IP address you are using.
* It COULD do all sorts of profiling and statistics to make sure it’s the same browser that is requesting all the subsequent packets
* However, it DOESN’T do these things and only relies on sequence numbers and port numbers to glue everything together
* The weaknesses above, allow for an attack called a **Session Hijack** because these conversations are called sessions.
  + - Someone else can hijack your connection / session if they can jump in and persuade the server to send the next packet to them and increase the sequence number.
    - And if that happens, the source will still be trying to connect to the destination, but as far as you’re concerned, the source is still trying to connect to the destination but the sequence number to the destination has gone way up so it will keep replying to the source asking wtf are you saying, that’s an error.
    - The source will end up timing out, but someone else will have picked up your session and keep going.
    - E.g. You are on Amazon, typing in your credit card details and you’re about to buy something. Suddenly, Amazon stops responding. Someone over in Russia is hijacking your session, has your credit cards, your login and they’re going around buying things.
* Shim attack situation
  + Shim had his computers all set up, relying on the R authentication protocol, which allows multiple computers to talk to each other in a trusted way.
  + Using the Unix R setup, Unix will let you log in as a trusted person as long as you’re coming from a trusted computer.
  + So, if you’ve got six computers and you’re setting up the R-trust relationship between each of them, any one of those computers can just automatically do things as YOU on any of the other machines without anyone along the way asking for usernames / passwords. It will only ask you for a username / password to log into the first machine, but from that machine, you can connect to any other machine.
  + This is a crazy thing, because if someone compromises one of the machines, they’ve got them all.
  + You could say that, “but they’ve still go to compromise one machine”, except **how does authentication work?**
    - Suppose two machines are in a trusting R-relationship with each other, you send a message to this machine, saying to delete that file.
    - Since you trust all the machines that are connected, you will delete it.
    - But how do you know that its really coming from that machine?
    - **The authentication in the R-protocol was entirely done by the IP-address of the packet, the source address.**
    - So when you get a packet from someone, on that packet, it says where it came from and if it says it came from Richard’s machine, you’d trust it.
    - **It is hence not really well-authenticated because anyone could LIE or SPOOF / spoof the source address**.
    - You can just send a packet with a different source address inside it.
    - The whole R-protocol is subverted now, however there is a problem.
    - R lives at the level of TCP and TCP is not IP, it doesn’t work on individual packets struggling around, it works on conversations. For this whole thing to work, we have to have done the handshake and set up a conversation.
    - Say a source with IP address 1.1.1.1 wants to connect with Wikipedia  
      For the attacker to successfully do a session hijack, they must do a handshake with Wiki
      1. The attacker sends a packer to Wiki saying that they are 1.1.1.1 and a SYN
      2. Wiki replies with a SYN-ACK, sending it to the real source 1.1.1.1
      3. The real source says wtf I didn’t ask for a connection? They will drop the connection.
    - From above, an attacker can’t do a three-way setup / handshake because part of it is the Wiki needs to respond to the attacker and then the attacker responds back to set the connection up. However, Wiki would be responding instead to the real source instead of the attacker so this won’t work.
    - This was the clever method that someone thought of, since spoofing an IP address won’t work as a handshake can’t be made.
    - However, there is a way to get around this.
  + **Attackers can get around this by working out the session number between them and just send the packet with the session number**. Here is how it happened in the Mitnick / Shim attack:
    - Christmas 1994
    - Two trusted machines + Mitnick’s machine.
    - Keep requesting packets / DDOS Shim. Set up a whole lot of half-open connections.
    - Shim can’t take anymore packets, because his machines are full-up / are waiting for connections that haven’t been closed off.
    - Shim’s machines are now stuffed.
    - When the destination sends Shim a packet, Shim’s machines won’t respond with a reset because his machines are DDOS’d.
    - Mitnick sends the destination a packet, with a spoofed IP address to ask for a connection / **SYN**
    - Destination, responds with **SYN-ACK 1.1.1.1** and sends it to Shim, but it gets lost because his machines are dead/stuffed.
    - **Mitnick didn’t get the SYN-ACK, but PRETENDS that he got it and reply with an ACK**.
    - Mitnick will always know the sequence number being sent TO the destination, however still won’t know the sequence number sent from dest 🡪 Shim’s machine.
    - Mitnick very very quickly connected to Shim a couple of times at the beginning and tried to establish a couple of connections with him / looked at the sequence numbers that he sent, which followed a very simple pattern.
    - Each new sequence number increased by 128,000.
    - Mitnick could now do something called **Sequence Number Prediction**. He could predict that the next connection that gets established with the destination is going to have a sequence number 128,000 bigger than the last one that has been sent from Shim to the Mitnick, as long as no one else is connecting to you. (probably not, since it is Christmas at midnight 1994. No one is probably even on the internet).
    - Mitnick sends Shim the half-connection 🡪 Shim replies to Mitnick 🡪 Mitnick knows the size of the SYN-ACK packet and knows the sequence number that he probably uses.
    - Mitnick can now just establish a connection with the destination.
    - **This attack relied on the fact that things weren’t authenticated properly.**
    - **Relying on an address / number that people tell you themselves as an identity is NOT a good way of authenticating someone + also relied on the fact that sequence numbers were predictable.**
    - After that, a lot of people have tried to put a lot of work into **randomising sequence numbers**.
* Session Hijacking is still a very popular attack
  + It is getting less effective now though, because people are making better and more random sequence number generators
  + However, you can think of all sorts of flavours of this attack that you could do.
  + E.g. Attacker says they are 1.1.1.1 and sends a message to the dest and the dest replies to the real source, the dest might not be replying directly to the source, but the packet is bouncing around the internet before it gets to the source, so if the attacker manages to take over one of the machines along the way, they can observe the packet as it goes past and see the sequence number.
  + Generally, sequence numbers is a flawed way of authenticating someone assuming that attackers won’t be able to predict the numbers and allowing the attackers to self-authenticate.
* Also, going back to timing of an attack, the best time to attack is when no one is expecting it. When they are distracted, or a company is closing down, or there is a public holiday.
  + These are critical times to attack because everyone is **misdirected** at the time.
  + The normal level of attention that they have, is being put on the trip that they’re imagining they’re about the have etc.
  + This is the really clever part about the attack.
  + Make an alarm go off many many times in a building, so that no-one will listen / pay attention to it anymore.
* **When a system goes nuts and is not doing its job, the natural tendency when everyones in a crisis is to just turn down the security, because if you turn down the security, you are turning up the exposure to risk.**
* **With risk, it is INVISIBLE.**
* **You can increase your company’s exposure to risk and no one will know, unless you’re unlucky.**
* **You could be engaged in all sorts of bad practices and no one will find out because you’re changing your risk from 1/1000 to 1/100. You’re probably still safe**
* **Airport security can turn down the security, to lower chances of innocent people being stopped from bad facial recognition and annoying everyone etc. but by turning this down, you are silently turning your risk upwards.**