MORNING: Richard is via Skype

* Initialisation vectors (IV’s): when an algorithm deals with a sequence of things, e.x. Merkel-Damgard: we don’t have something to start it off, so it’s the first block of data that we use to start of the hash
  + We do this to protect the whole document, with the avalanche effect
  + A tiny bit change can completely change a document
  + Whole hash with a block, roll on, hash with next block
  + When you come up with a password, due to the lack of entropy, you often don’t get as many bits of security as you think
    - Most people don’t remember a 10 letter completely random sequence of letters, so they take the first letter of a phrase
      * E.x. from the national anthem
      * But this is flawed, dictionary attack could include all the famous phrases from films, bibles and such so the bits of security then are still smaller than you expect
    - WEP had a 64-bit key, a decent amount of work, for the random sequence generator for the XOR
      * How do we make people make a 64-bit password?
      * WEP used an initialisation vector to fill the bits
      * 24-bit initialisation vector
      * If the same data gets encrypted with the same key each time, you can recognise patterns in the plaintext
      * So to get a collision, you get sqrt(2^24) = 2^12 packets sent
  + So IV’s try to add entropy but they usually lead to trouble if attackers know where in the bit stream they are located, can detect patterns and changes
* Mixing Data and Control story:
  + Post office – fill the satchel with things and get the post office to post you your satchel, then you don’t need to take the goods out yourself
* Smashing the stack for fun and profit: computers are always context switching. When they get interrupted, do something else, then goes back and such: it is always rapidly doing changes.
  + We go back to where we were before the interrupt
  + These jobs are kept track of on the stack -> the top of the stack is what gets reverted back to after it was interrupted
  + Buffer overflows: when a process is actually being executed, it is at the top of the stack. When it is finished, its data is thrown away and the stack pointer goes to the next process to wake it up.
    - Someone needs to fill this woken up process of its state -> it’s stored on the disk and is read back and loaded
    - The process that’s running can write to the stack -> then store data on the stack so we don’t read from the disk each time
    - So, we write to the stack above the data for the sleeping processes and grows down towards LARGE addresses
    - So, if you can override your address space, you can write on the data in the stack frame for a sleeping process
      * Including a pointer to the instruction that the sleeping process was going to do next
    - Override the return address, so it points to an area of memory you control, then you make it execute the code that you wrote
  + LOOK UP BUFFER OVERFLOW ATTACKS (stack, return address, & how to do it)
  + <https://www.youtube.com/watch?v=1S0aBV-Waeo>
    - So, if you’re writing code, and take the input from a user of length 40. The user puts their code in array, write a nop sled, and once they get to the return address of the next process, make it point to the start of the array so it sleds to the malicious code
* Proof of work: friction of someone having to do a lot of work in order to prevent them from doing an attack
  + We make someone do a lot of work (key stretching)
  + Proof of work on bitcoin: we make it a lot of work to make a new page in the ledger/alter a page in the ownership ledge.
    - Blockchain means if you change a page/block, you have to change all the other pages/blocks -> and do it before the next person solves a puzzle to add another page. Means attackers must do heaps of work to fake pages.
* Moore’s Law:
  + Moore observed that about every year or 2, the number of transistors in a chip doubles -> and the compute power/speed also increased
  + Compute power roughly doubles every 18 months -> so every 18 months, you lose a bit of security.
  + A random prediction that ended up being quite applicable
* Hard Disk encryption attack: found that if you tell Windows to encrypt the hard disk, you assume it is. But to save time on the encryption, Microsoft queries to see if the disk has its own encryption, and then just gives it the plain text.
  + So the disk says “Microsoft, don’t bother doing your encryption, me (the disk) will do it”
* Normally we don’t attack the encryption (front door) – usually exploit the backdoor/the way it was implemented
* Encryption algorithms: Ciphers after ww2 and pre-RSA:
  + RSA works because it’s hard to break the mathematics
  + Everyone was rolling their own, so NIST was asked to come up with a standard that everyone could use that was good enough.
  + NIST realised that no one knew the principles of good encryption – so the ran a competition for everyone to submit their own cipher algorithm, and all would be made public for other people to try and break them, and then they pick the best
    - The only decent thing they got was from IBM, a cipher called lucifer
    - They submitted DES (data encryption standard) & NIST chose it, and sent it to the NSA to see if they could help at all
      * NSA is very secret, no one knew how big they were
    - NSA looked at it and gave it back and said it was okay, and changed one of the numbers in the s-boxes from 17 to 13 and DES was published
      * The NSA was discovered to be on a whole other level, since the s-boxes had random numbers, NSA changed 1 number
        + Turns out NSA knew about differential analysis decades before cryptographers, which prompted their 17->13 change which caused heaps of discussion about the need to know more about this field, amongst cryptographers
    - DES: jumbles, Feistel cipher -> uses an internal function called f, which doesn’t have to be invertible (new feature) which is very clever
      * Wasn’t enough bits, could be brute forced
  + New competition lead to the new standard: advanced encryption standard **(AES)** -> uses substitution permutation network
    - Does substituting, does permutating, jumbles things up with a key and then iterates that key
    - Does a lot of complicated work, makes it hopefully impossible to undo
    - Has been broken, but not in any feasible ways

EVENING LECTURE HANDWRITTEN NOTES/BLOG ON TUMBLR