COMP6441 Lecture 10

Youhan Cheery

May 2017

1 Miscellaneous

- Mixing data with control (send someone an email and tell them to do something). Data and control are intertwined at every level of abstraction.
- Getting control of data can provide a malicious hacker with a great deal of power. Changing data changes behaviour of the system, e.g. with self-driving cars. A machine that can be affected by changes in data is a hackable system, "a change in x data, results in y outcome".
- To authenticate we use a shared secret.
- To prevent repeat attacks we use a nonce.
- Authentication is fine when you have a shared secret, but how do we have secure communication with someone we don't know? We need to work out a shared secret on-the-fly.
- Spoiler: there's no real good way of doing this.
- Horton principle: when you validate something, you need to make sure you are validating/checking the right thing. What you say needs to be what you mean, and vice versa. You may think you are authenticating for one person, but in reality you are authenticating something else.
- Deep packet inspection - everything on the firewall, even encrypted, is read. Compromises certificate store, and allows anyone to compromise their product.
- Forward secrecy: if a listener is watching encrypted traffic, and eventually learns the shared secret, they could go back in time and study previous messages. How to prevent past messages in the case of future compromise: perfect forward secrecy.
- Perfect forward secrecy can involve setting up an RSA session at the beginning of the conversation. In the messages, the key is given. This is the 'session key'. Every conversation has a key for its session. At the end of the conversation the key is thrown out. Problem: if someone learns the setup conversation, they could crack the protocol.

- Another way of doing it is through a public key channel (slow, expensive) where the parameters of Diffie-Helman are agreed upon and used for the conversation.
- Another way: in that session send a disposable public key, which is used to encrypt the session key. Once the session key is attained the public key is thrown out.
- DNS spoofing - when something comes in, tell switch to redirect incoming traffic into the switch that you want. Everything trusts everything at this layer of the network.

2 Sharing keys

2.1 Centralised

- Centralised authority. Asking a 'bank' of secrets and asking for the secret. Problem: single point of failure. E.g. PKI
- To talk to someone you use peoples' public keys. Susceptible to man in the middle attacks. PKI solves this by constantly timing out.
- How do we set up this trusted worldwide bank that hosts all these public keys?
- Certificates (x509) let us know we can trust someone.
- CA = Certificate Authority. Web browsers come pre-loaded with the public keys of CAs, which can then 'issue' trust.
- Signing is like giving your permission onto something. This is basically like encrypting something with your private key.
- x509 has a revocation implementation - meaning that browsers can revoke certificates.
- See Lenovo certificate issue.

2.2 Decentralised

- Decentralised approach: peer based web of trust. The things that people you trust think make up your ability to trust.

2.3 Block Cipher Modes

- Lots of ciphers work on blocks: helps with avalanche, diffusion properties as you have more data to make a decision with.

- Block ciphers (such as AES) break a message into chunks of fixed size. Each chunk is encrypted and then the chunks are combined to create a ciphertext.
- How they create the blocks and combine them is known as ‘block cipher mode’.
- ECB = Electronic Code Book. Encrypts each chunk independently and concatenate them together. This is the simplest way of doing it. ECB is bad because for a big plain-text, there is a large probability that there are repeats of blocks. This allows you to detect and then decrypt. ”Fuzzy penguin” - linux penguin that encrypts to the same thing because it has so many common features.
- In the message if there are two plain-text chunks that are the same, you’d want them to map them to different things.
- Counter (CTR) and Cipher Block Chain (CBC) are the other modes.
- Padding method: PKCS7. Not designed so much for cryptography but does find use.
- In CBC ciphertext gotten from the first block is XOR’d with the plain-text of the second block, which is then encrypted. Problem: You need the previous block before the next one in decryption, which means if you lose a block you can’t continue the decryption. Positive is that you can decode in parallel because you have the entire cipher-text.
- In CTR mode what you fold in each time is not the previous cipher-text, it’s a nonce (counter). You’re encrypting the nonce which is then XOR’d with the plain-text. Can encrypt and decrypt in parallel.
- Both CTR and CBC are designed to prevent the repeat of cipherblocks that ECB is susceptible to.