CSCI 5271 Exercise Set 3

Alex Biedny

November 5, 2020

Problem 1. Protocol Droid

(a) The simplest way that E could fool C is by first sending a TIE fighter to record a conversation between A and C. Then when E wants to impersonate A, E can start the protocol by sending a previously recorded t-bit string N_A and the recorded $MAC_K(N_A)$. Because C does not check if N_A was previously used, the recorded message will be successfully verified by C, and E will have impersonated A.

(b) This makes it a bit more complicated for E to impersonate A, but doesn't really change anything. When E wants to impersonate A, he needs to know a random string that has not been used for N_A before, and the $MAC_K()$ of that string. From a previously recorded conversation, E can use an intercepted $MAC_K(N_A)$ as the N_A, and use the $MAC_K(MAC_K(N_A))$ recorded from the same conversation to be the $MAC_K(N_A)$. This way E has a random string, and a valid $MAC_K()$ of it and can impersonate C.

Problem 2. Revoking Certs

(a) If someone is unable to contact the CA of a certificate, there is no way to know whether or not that certifcate has been revoked, and thus no way of knowing whether or not you are talking to the real person, or an adversary impersonating them. As it relates to availability, this would mean that a constant line of communication to the CA would be needed to verify a certificate. If this line of communication went down or was disrupted, there would be no way to verify the certificate, causing an availability issue.

(b) This doesn't change the previous availability vulnerability: There still needs to be a line of communication to and CA. While it would be more difficult for an adversary to disrupt communication to all CAs, it would not be impossible.

In addition, this creates a few other problems. If every CA stores revocation lists, then if a single CA is compromised, they are able to revoke any certificates, no matter who originally signed it. This also introduces problems if there are ever mismatches in the revocation lists between different CAs. In the case of a certificate being revoked, it also creates the need to notify every CA of a revoked certificate, rather than just one.

(c) The most obvious issue with using the compromised public/private key pair is that if the private key is compromised, anyone could impersonate you and tell the CA to revoke your certificate. While this should happen anyways if a private key is compromised, an adversary revoking a certificate without your knowledge could lead to availability problems, or could be used as part of a more sophisticated attack.

Problem 3. More Signatures

First, this does not solve the low water mark problem. It may move that water mark up, but the system is still only as secure as the weakest n number of CAs. In addition, this model has several problems with it. In the real world, getting signed by a CA usually boils down to paying them, and if certification is required by n CAs, then the cost of getting a trusted certificate goes up. This also introduces the problem of verifying those certificates.

In the current system, operating systems trust root certificates, and these form the base of certificates chains. Operating systems are always able to follow the certificate chain to its source and verify it using one of the root certificates that the OS trusts. In the web of trust model, there would still need to be some concept of a root certificate that the OS implicitly trusts, as an adversary could easily create n CAs to sign a malicious certificate. With no way to trust a CA signature, the number of them signing really doesn't matter.