Project 1 :: Defeating SkyNet (Security Essentials)

Dean Pisani
Kristy Hughes

April 25, 2014

1 Diffie-Hellman

We used a 3073-bit safe prime because it is a larger prime than the skeleton code, hence harder to factorise. We added a variable called prime generator so that the developer can change it from the default of 2. We set it at 5 so that the public key is larger and harder to crack, but it is generally set to 2, 3 or 5. The private key is a random number between 0 and the prime. It is better to have a large set of numbers that this could be so that it is difficult to guess, however it is pointless making it bigger than the prime because all arithmetic is done modular the prime. Our public key is the standard Diffie-Hellman public key based on the private key and the prime.

Parameters:

- Raw Prime: raw_prime is a 3073-bit safe prime from RFC 3526
- Prime Generator: $g = 3$
- Private Key: Random number between 0 and the prime
- Public Key: $g^{\text{private\_key}} \mod \text{raw\_prime}$

```python
# raw_prime is a 3073-bit safe prime from RFC 3526
prime = read_hex(raw_prime)
# Choose a prime generator, usually 2,3 or 5
g = 3

# Creates a Diffie-Hellman key
def create_dh_key():
    # Choose a random integer between
    # 0 and the prime number as the private key
    private = random.randint(0,prime)
    # The public key is is the
    # prime generator to the power of
    # the random private key mod a prime
    public  = pow(g, private, prime)
    return (public, private)
```

2 Cypher

We chose AES as our cypher because although TDES is cryptographically secure, it is not as fast or as secure as AES. AES is now industry standard thus we thought it best to use it.

3 Tampering Prevention

To prevent tampering of the message we use data origin authentication. To do this, we implement HMAC which hashes the message using the sender's MAC address as the key. Since a MAC address is unique to the user the attacker is unable to replicate this or tamper with the hashed message.

4 Replay Attack Prevention

Replay attacks occur when an attacker takes a previously sent encrypted message and sends it to the server again. To prevent this, the user appends a timestamp at the beginning of the message and when the server decrypts it, it checks that the timestamp of the current message is later than the timestamp of the previous message recieved.

5 Peer to Peer file transfers

6 In the Real World