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: $g^{\text{private\_key}} \mod \text{raw\_prime}$

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

```
4  g = 3
5
6  # Creates a Diffie-Hellman key
7  def create_dh_key():
8      # Choose a random integer between
9      # 0 and the prime number as the private key
10     private = random.randint(0,prime)
11     # The public key is is the
12     # prime generator to the power of
13     # the random private key mod a prime
14     public  = pow(g, private, prime)
15     return (public, private)
```

2 Cypher

3 Tampering Prevention

4 Replay Attack Prevention

5 Peer to Peer file transfers

6 In the Real World