ELEC5616 : Assignment 1
SKYNET Design Document

V. Grotte 309193168

What was your choice of Diffie-Hellman key exchange parameters and what made you select them specifically?

I chose the “2048 bit safe prime for Diffie-Hellman key exchange obtained from RFC 3526”. The generator that is recommended for use with this prime is 2. I created each private key with PyCrypto.random.randint(2, (2^8)). I created each public key from the calculation : pub_key = (2^private_key) mod big_prime.

My reason for bumping up the size of the prime was the “Logjam” vulnerability [1][2]. The vulnerability allows keys with sizes of 512 or 1024 bits to be cracked effortlessly by leveraging the knowledge that keys were used repeatedly and that the procedure of generating key material with those primes was performed in a series of steps, some of which could be precomputed. It seems like an example of “tomorrow’s computers not keeping today’s secrets” but I’m not certain. I didn’t make the prime larger because, even if it would have made my key material super secure, it would have been a pain to test and implement and also key material is not the only attack surface in a system so it would be inefficient to spend too much time implementing the key-exchange only.

What was your choice of cipher? What mode of operation does it use?

I chose AES for my cipher because it seemed like a solid standard. It’s mode of operation is CFB or Cipher Feedback. This type of cipher extends the block cipher concept into a self-synchronizing stream cipher. For my cipher’s key I used a SHA256 hash of a portion of my Diffie-Hellman shared secret (ZZ).

I chose CFB because according to stack overflow [3] while it is a second best to OCB in terms of code convenience and security, OCB seemed to have some patent issues surrounding it and therefore CFB seems to be the norm compared to the dozen or so other modes, most of which seem defunct or to have some specific niche purpose.

For the authentication process I implemented a RSA public-key cryptosystem using the standard “Optimal Asymmetric Encryption Padding” RSA algorithm (PKCS1_OAEP) to produce the public/private key pair, the purpose of doing this is to provide better security for the key material used for the cipher that encrypts the actual communication messages.

How do you prevent attackers from tampering with messages in transit?

SHA256 hashed message authentication codes (HMACs) are appended to every package that is sent between bots except during the initial authentication procedure, where a more complex scheme is used to attempt to ensure the integrity of said authentication. An initialization vector is spawned using Random.new() and prepended to the (DH) public key. The IV in this case is used to verify message integrity. The initial broadcast message from the bot is encrypted and both messages in the next back and forth exchange are signed and verified using the Pycrypto PKCS1_PSS implementation.

How do you prevent replay attacks?

The CFB system provides protection against replay attacks as the cipher relies upon generating a unique, random session ID in the form of an IV and each block that is encrypted uses the previous material as input into the cipher function. This makes it much more difficult for someone to compromise a session as the IV changes each time.

Why might we want to allow for peer-to-peer file transfers between bots? What are the advantages and disadvantages to using a central web server (pastebot.net in our case, similar to pastebin.com) to distribute files when controlling a botnet?

Giving the botnet P2P functionality means bots can communicate to each other during the execution of tasks without having to go through a central server. This is referred to as a distributed architecture and vastly increases the power level of a swarm.

Some potential advantages to a central web-server architecture:

- Simpler to implement
- Communication doesn't have to "propagate" through the network

The comparative disadvantages are:

- Single point of failure.
- Easy target to track.
- Easy target to break.

Explain how this botnet, if used in the real world, could be trivially controlled by other hackers and government agencies. How might one attempt to stop it?

The botnet has addressed much of the security requirements for its communication channels, but there are still many vulnerabilities with regards to the environment and operation of the program :

- It doesn't obfuscate its code or its presence in a system at all.
- It takes no steps to secure or control the OS it is running on.
- The program itself has no authentication and stores keys in plain text in the shipped code.
- It has one root control server.

Each of the above points introduces an attack surface into the botnet. For example, having one central control server makes for a simpler target (compared to a distributed design) for anyone who wants to steal the botnet. Especially since, in this implementation if a node is compromised and it has nodes beneath it, then an attacker also has that sub-tree.

[1] https://security.stackexchange.com/questions/129885/why-are-primes-reused-in-diffie-hellman-key-exchanges

[2] https://en.wikipedia.org/wiki/Logjam_(computer_security)

[3] https://stackoverflow.com/questions/1220751/how-to-choose-an-aes-encryption-mode-cbc-ecb-ctr-ocb-cfb

[4] https://weakdh.org/

[5] https://stackoverflow.com/questions/40333804/encrypt-text-with-pkcs1-oaep-python

[6] https://stackoverflow.com/questions/1631727/links-for-aes-128-bit-cfb-implemenation-or-sample-application

[7] https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Cipher_Feedback_(CFB)

[8] https://en.wikipedia.org/wiki/Optimal_asymmetric_encryption_padding

[9] https://msdn.microsoft.com/en-us/library/cc750036.aspx

[10] https://en.wikipedia.org/wiki/Replay_attack