

Defeating SkyNet: Design Document

Stephen Tridgell - 309205867, Nicholas Carydis - 310183626

Botnets have had a prominence in the media over recent years, effectively instilling them into the public consciousness as another threat to the security of computer networks. The security protocols of an established botnet, although rarely addressed by the media, are an interesting and informative way of looking at the threat and of determining ways of having it neutralised.

The first part of the project involved securing an established botnet, by developing protocols to provide authentication, confidentiality, integrity and replay prevention between infected hosts. Design choices made and the reasons for them are outlined below.

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

The parameters chosen were those listed as part of the 2048 MODP group of the IETF standard (RFC 3526). The default allocated prime was increased in bit size to 2048 bits. The larger prime was selected to make a brute-force attack less feasible via the avenue of the discrete-logarithm problem, for a negligible cost of resources. This did not, however, increase the strength of the resultant symmetric key, as the shared secret is hashed using SHA256 to produce an un-biased 256b symmetric key.

The generator (or 'g') for this prime is 2, and the PyCrypto secure Random function was used to generate a random 'a' and 'b' for the two parties.

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

The AES cipher was chosen with a symmetric key size of 256 bits. This key was produced by hashing the shared secret produced by the Diffie-Hellman key exchange. The choice of key size was based on the largest possible key supported by AES, so as to provide the maximum security possible at this juncture. The cipher was chosen to run in Cipher-Feedback mode (or CFB) so as to perform as a stream cipher, for the issuance of commands and reception of data between infected hosts.

For authentication purposes, a public-key cryptosystem (RSA) was used. PKCS1_OAEP (Optimal Asymmetric Encryption Padding) was used for the production of a public/private key pair, which was the standard RSA algorithm combined with a padding scheme to mitigate the risks of frequency cryptanalysis on the resultant ciphertext.

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

Our botnet is designed to be implemented in a tree-like structure. Each parent node has a private key and the child has the public key. The child acts as a server for the parent node. When the parent attempts to connect to the child an RSA protocol is used to authenticate the parent. When the client (parent) sends a request to a server (child), the server replies with a challenge. This challenge includes a random IV along with one half of the DH secret (g^a). This challenge is encrypted using the client's public key (known by

the server), providing confidentiality. A potential attacker can masquerade as the child node in this instance, however this does not compromise the botnet. The client replies with (g^b) along with the IV, which it signs with its private key. IV and (g^b) will be public in this case, but the digital signature will ensure message integrity. The public disclosure of IV and g^b will not compromise the security of the system as demonstrated by the Diffie-Hellman algorithm. The server verifies the signature and accepts the connection.

For other commands and data, the data are appended with 8 bytes of hash, which is then encrypted using AES. This provides confidentiality and integrity.

- **How do you prevent replay attacks?**

In the authentication stage, the random IV provides protection against replay attacks, as it changes every time the authentication protocol is invoked. For other communications, the use of CFB prevents replay attacks as a) the CFB system begins with the IV (which changes with each new connection) and b) future cyphertext outputs depend on the encrypted value of the previous blocks of cyphertext (for which an attacker would need to use the key to predict). Thus sending the same block of cyphertext twice will not decrypt to the same plaintext.

- **Why might we want to allow for peer-to-peer file transfers between bots?**

Using p2p file transfers allows the bots to run in a distributed fashion, without a need for a central controller. The advantages of using a centralised system are generally the use of less complex authentication mechanisms, as well as more judicious control over the bots in the network. Instructions and data would not require propagation through a hierarchy of hosts, and so would reach their destination faster too.

However, the disadvantages are numerous: It produces a single point of failure, and would pose an easy target for governments and other hackers to take down or commandeer the botnet. It would be much easier to trace commands emanating from a single web server, and for that web server to be taken down – either by redirecting its traffic at lower levels or DDOSing it. The server itself may not be able to cope with capacity demands as more and more hosts are infected and become part of the botnet – it may end up DOSing itself.

- **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?**

As mentioned, this centralised botnet could be trivially controlled once the central web server is compromised. The modifications we have made (including decentralising the structure into a tree-like hierarchy) defend against some of these attacks and make the root node harder to trace. However, even in this structure, a compromised node will effectively compromise the child nodes beneath it, and compromising the root node will compromise the entire botnet.