Part 2 :: Protecting the Castle (Commanding the Legion)

Dean Pisani 311210775
Kristy Hughes 310186293

1 Authentication

How do you ensure the only one who can send updates to SkyNet is the botnet master?
Updates made to pastebot.net need to be signed by the master for them to be considered valid. We used asymmetric encryption using the RSA digital signature protocol with appendix according to PKCS#1 PSS, which counters vulnerabilities introduced by using RSA encryption without proper cryptographic padding. The RSA keypair generated by the master is used for this signature, since the signatures make use of the master's private key. A bot then uses the master's public key which is made available to verify the signature. Only valid signatures are accepted by the bots, and since signatures can only be made with the private key the master holds, this ensures only the master can send updates to the bots.

```python
def sign_file(f):
    # Read in the private key
    key = RSA.importKey(open(os.path.join("TOP_SECRET_KEYS", "master_rsa")).read())
    # sign using the RSASSA-PSS scheme
    # use a SHA-1 hasher for use in the RSA signing object
    h = SHA.new()
    h.update(f)
    signer = PKCS1_PSS.new(key)
    # create the signature and prepend it to the message
    signature = signer.sign(h)
    return signature + f
```

2 Confidentiality

How do you protect the valuable information to ensure it can only be read by the botnet master?
Since the valuable data obtained by the botnet is stored on a publically available server, pastebot.net, it is nessecary to ensure confidentuality. To ensure this, bots first encrypt all valuable data using RSA and the botnet master's public key. With an asymmetric cipher like RSA, we can ensure that any valuable data cannot be read by anyone - including bots - since they do not know the master's private key. The only way to decrypt these encrypted documents is using the master's private key, either by stealing it from him/her directly (which may be difficult if it is password protected and hidden) or by factorizing the public key (which from wargames we have seen is very difficult).

Here is the encryption code that any bot can perform using the master's public key

```python
def encrypt_for_master(data):
    # Create a SHA-1 hasher for use in the RSAES-PKCS1-v1_5 encryption process
    h=SHA.new(data)
    # Read in the master'spublic key
    key = RSA.importKey(open(os.path.join("pastebot.net", "master_rsa.pub")).read())
    # Create a cypher object with the key
    cipher = PKCS1_v1_5.new(key)
    # Encrypt the valuable data
    ciphertext = cipher.encrypt(data+h.digest())
    # Return encrypted data
    return ciphertext
```

Here is how the botnet master (or anyone with access to the botnet master's private key) would decrypt valuables

```python
def decrypt_valuables(f):
    # Read in the private key
    key = RSA.importKey(open('TOP_SECRET_KEYS/master_rsa').read())
    #Get the SHA-1 hash's digest size
    dsize = SHA.digest_size
    sentinel = Random.new().read(15+dsize)
    # Create the cypher object with the private key
    cipher = PKCS1_v1_5.new(key)
    # Decode encrypted data
    decoded_text = cipher.decrypt(f, sentinel)
    # Grab the digest and if the digest is right, print the decrypted data
    digest = SHA.new(decoded_text[:-dsize]).digest()
    if digest==decoded_text[-dsize:]:
        # Remove the digest from the message
        decoded_text=decoded_text[:-dsize]
        decoded_text = str(decoded_text, 'ascii')
        print(decoded_text)
    else:
        print("Bad encryption")
```

3 Integrity

How do you ensure the botnet updates signed by the botnet master cannot be forged or modifed?
The digital signature ensures integrity also. If an attacker were to attempt to create their own updates, or modify one in transit, then the signature will not be valid, since it has to be created using the private key combined with the original message. For an attacker to succeed in creating their own update, they mus either have access to the botnet master's private key, or be able to factorise the public key.

4 Breaking into Skynet

If SkyNet's botnet code is dismantled and/or the source code for it stolen, does your scheme become less secure?
There is no cryptographic information stored in the source code other than the type of encryption being used, and so if the code was dismantled or stolen it would not compromise the integrity. The security of this encryption method and signing method is in the strength of RSA - the fact that to decrypt the data, one must need the master's private key or factor his public key.
If one of the bots were to be infiltrated, the attacker would not have access to any of the master's private data, since the bots only possess the master's public key. Thus, it would not give the attacker any means to disrupt the rest of the network, since they still cannot sign messages. They also cannot access any of the data uploaded by the other bots, since that also requires the private key. However they would have control over that particular bot, and could use it for their own means. Potentially they could use the public key to create fake information that will be still accepted by the master, giving them other means to attack.

Give an indication of how difficult it would be for an adversary to take control of SkyNet when your protections are used.
It would be very difficult for an adversary to take control of Skynet. An infiltrated bot by itself would only be able to add to the valuables owned by the botnet by uploading encrypted data it discovers. It would not be able to obtain valuables mined by any other bot, since each bot encrypts it with the master's public key before uploading. Since the encrypted files require the botnet master's private key to decrypt, the infected bot would not be able to infiltrate further or gather useful information. So the aim of the infiltrator would be to take control of the botnet master. And this would be difficult, since there is no indication of which bot is actually the master in a huge peer to peer networkSo the problem of infiltration is reduced to the problem of obtaining the botnet master's private key. Thus it is very important for the botnet master to keep the private key in a secret place away from the reach of any bot and possibly encrypted using a password.