Project :: Defeating SkyNet
Part 1: Security Essentials

ELEC5616
Luke Anderson *luke@lukeanderson.com.au*

11th April 2018

Final due date is yet to be determined

Introduction

It's 2018. Almost every device with a CPU in it has been connected to the Internet. Whilst this is a stunning advance for humanity, the security for these devices has come as an afterthought or not at all. Millions of computers and devices, all with valuable information and processing power, are left vulnerable to attack.

Blackhats, and even possibly governments, have created viruses, worms and other dastardly schemes to mine for information and turn a profit using these weaknesses. In this project, we'll be specifically looking at botnets: how they work, why they're valuable and why it's so difficult to defeat them.

Botnets perform various tasks including but not limited to:

- Stealing confidential information (passwords, banking details, etc.)
- Sending spam email
- Distributed Denial of Service (DDoS) against chosen websites
- Mining for Bitcoins
- Hold files for ransom by encrypting them and charging for decryption
- Providing a secure proxy network for other illegal enterprises

Of course, such a valuable network is not likely to go unnoticed for

long. You'll be having well funded organisations, government agencies and other hackers attacking you. For that reason, SkyNet will need advanced cryptography to ensure your blackhat plans can't be stopped.

Background & Disclaimer

This project has been created to help you gain an understanding of how practical a massive cyber attack is and how complex it can be to defend against it. The basic technology of this project is pulled from the Conficker worm that ravaged the Internet in November 2008. At its peak, Conficker controlled up to 7 million computers across 200 countries. Conficker would only have needed 2 million of those machines to overpower the top 500 supercomputers at the time combined. To combat the threat, Microsoft formed an industry group to counter Conficker, composed of numerous security and technology companies. This group also conversed with government agencies around the world.

To understand why it was so difficult to slow down or defeat, we'll be implementing key components of this botnet that utilise advanced cryptographic techniques.

This is not an operational botnet nor do we intend you to create one. To defeat blackhats, you must understand how they work and the techniques they use. Recent botnets have used advanced computer science and cryptographic methods in order to remain secure from both blackhats, whitehats, well funded organisations and even governments. These advanced methods are what we intend you to learn and what we believe will give you the skills to detect, prevent and disassemble such attacks in the future.

Part 1 :: Securing the Channel

When you're transferring secrets, be they banking details or Bitcoins, you don't want to be overheard. Additionally, communicating in the open makes it easier for SkyNet to be detected via network analysis. Botnet authors don't like easy ways for computer admins to pinpoint infected machines.

In Part 1, you will need to:

- Implement key exchange using the Diffie-Hellman algorithm, when peer-to-peer connections are made between bots.
- Achieve confidentiality through encryption of the client-server communications with an appropriate block or stream cipher.
- Enforce integrity through the use of a MAC appended to all messages.
- Implement resistance against replay attacks using a mechanism which you are to devise.

1 Key Exchange

In order to strengthen SkyNet's communications against eavesdropping, you are to implement a method of key exchange for each possible connection. If you leave your communications unencrypted, it would be trivial for network analysis to indicate which machines are infected with your bot. It would also be easy to steal secrets and confidential information you might be sending back and forth.

For this project, you'll be using the Diffie-Hellman key exchange method. You'll be implementing this yourself from two standards created by the Internet Engineering Task Force (IETF). RFC 2631 describes how Diffie-Hellman works and how the calculations are performed. RFC 3526 provides standard parameters for use in Diffie-Hellman key exchange and an estimation on the strength they provide for computing a shared key. A shortened version of each of these RFCs has been provided with the source code.

The key that results from this key exchange should be used to compute all the other settings, such as the the block cipher keys or seeds for the stream cipher. Remember that you should hash the secret and use it as the seed for a random number generator instead of using it directly.

Note: Each and every session should use a different key. This means a

new key exchange session must be run every time a new connection is made.

2 Confidentiality

The confidentiality of the channel should be achieved through encrypting each message sent using either a block cipher or a stream cipher. An appropriate mode of operation for the cipher must also be considered. You may use any block or stream cipher you wish, provided it would remain reasonably secure against both government agencies and other hackers. The initialisation vector (IV) and key must be derived through a key exchange mechanism as described above.

3 Integrity

Message integrity is to be achieved through appending a MAC to each message sent across the channel. This is to help prevent any active attacker from modifying messages whilst in transit. The key to the MAC must be derived from the key exchange. You may use any MAC that you wish as long as it provides adequate level of security against any potential attackers.

4 Preventing Replay

You must devise a scheme where SkyNet is resistant to messages being replayed by an active attacker. The exact mechanism by which this occurs is up to you.

Imagine your bot could be told to perform a denial of service against a website by sending the message [DDOS www.ebay.com]. If it was trivial for others to resend that same message, others would be able to hold EBay to ransom as they could control your botnet.

5 Implementation

The bot's code is out of your control, being run on unknown computers that may in fact be owned by hackers or governments. You may assume the

command and control server's code is hidden, but remember, security can be reverse engineered.

An insecure skeleton framework written in Python 3 has been provided for you as a starting point. If you wish to use another language, such as Java with the Java Cryptography Extension (JCE), you may do so after seeking permission from your tutor. We can not provide technical support if you select another language however.

Note: This code has been written for the purposes of teaching cryptography and computer security. It is to be used as a demonstration only. No attempt has been made to optimise the source code.

6 Documentation

You are to write a 2-3 page design document outlining the security you implemented with your system. You should justify the choices you have made for your implementation and specifically discuss:

- key exchange
- authentication
- confidentiality
- integrity
- replay prevention

You should aim to address these questions:

- What was your choice of Diffie-Hellman key exchange parameters and what made you select them specifically? Make reference to RFC 3526.
- What was your choice of cipher? What mode of operation does it use? Why did you make these choices?
- How do you prevent attackers from tampering with messages in transit?
- How do you prevent replay attacks?
- 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?
- Explain how this botnet, if used in the real world, could be trivially

controlled by other hackers or government agencies. How might one attempt to stop it?

Your code must be well commented and in neat order.

7 Code Checklist

- Select parameters for and implement Diffie-Hellman key exchange in `dh/__init__.py`
- Select and implement a stronger cipher in `lib/comms.py` which will ensure:
  - confidentiality
  - integrity
  - protection against replay attacks

8 Marking

While it is important to learn how to write secure code and utilise security protocols, an equally important skill is the ability to read and analyse the code that others have written.

After the due date of this assignment, you will be asked to assess the code of 3 or 4 other groups, and the code that you submit will also be examined by a number of other groups[1], as well as being checked by tutors.

Your final mark will be comprised of:

- The quality of your report, as assessed by tutors and dedicated markers, who will look at:
  - Have you addressed the topics specified under section 6
  - Does the report demonstrate an understanding of underlying cryptographic principles
  - Is the report concise and professional, with appropriate use of academic language
- An assessment of your code by other class members, verified by tutors, assessing:
  - Does it work?
  - How strong is the implementation for each security goal?
  - Is the code commented and easy to read?
- How well you have assessed other class members
  - Code has been assessed
  - Marks given are in agreement with other markers (including tutors)
  - Relevant feedback is provided

Your assignment will be subject to plagiarism checking tools. While collaboration is encouraged, plagiarism is unacceptable and will be dealt with according to university policy.

[1] If you wish to have your code remain anonymous when being marked by others, don't enter your names within the submitted source code files

9 Submission Guidelines

Report:

Your report is to be submitted via the "Documentation Submission" link on e-Learning

Code:

Your code is to be submitted via phabricator:
https://answer.elec5616.com/
according to the instructions specified in the "Code Submission Instructions" (also on Canvas)