The Making of a Botnet

Yoray Herzberg

May 13, 2023

1 Chapter 0: Introduction

Hey everyone! This is a log of the making of a project of mine, a botnet. Hopefully, by the time you read this, this project will be completed. The log details the making of a botnet, which is a certain type of malware that can be used for DDOS attacks and remote control of many computers.

1.1 Section 0.1: Why a botnet?

You may ask, out of all the possible ideas for a project, why did I select a botnet? Well, there's no special reason for this, but my main field of interest is security, and more specifically, low-level security, web app security, mobile security, and malware analysis. I've had some experience with making my own malware for Linux and Windows (Mainly in C and C++. I have also had some experience with making Android malware), but they were not very complicated, as they were trojans designed to run on a single victim without any real C2 interface. I thought this project would be a nice way to advance my C skills, and also my malware analysis skills as we'll also add some anti-debugging mechanisms to this malware.

1.2 Section 0.2: The main design

This figure shows how the main architecture of a botnet typically looks:

Let us unpack this for a moment. There's a single C2 server that controls the victims of the malware (The victims can also be referred to as 'bots', hence the name botnet), and the C2 server can run commands on each of the bots and tell them to repeatedly send HTTP requests to other hosts, which can cause them to crash. For example, if we send a billion HTTP requests per second to a service, we may cause it to crash. Note that in real botnets, there may also be multiple C2 servers. This botnet will have 3 commands:

- Listing victims
- Running commands on a victim remotely

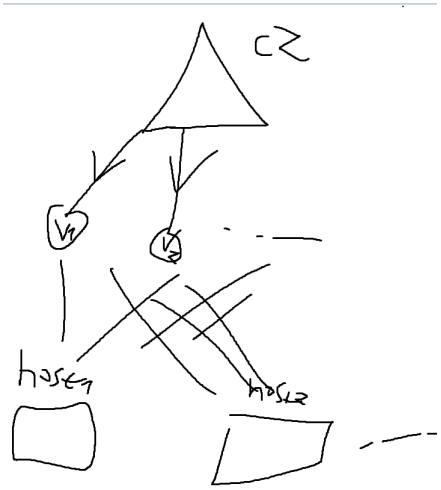

Figure 1: A simple botnet architecture

- Telling the bots to repeatedly send requests (e.g HTTP requests) to a given host

Chapters 1-3 will showcase the coding of each of these features in their respective order. In Chapter 4, we'll add anti-debugging mechanisms to the malware, and Chapter 5 will be a summary of the book.

1.3 Prerequisites

- Familiarity with the C programming language
- Not required, but knowledge of security will be very useful

2 Chapter 1: The C2 server

As previously mentioned, the first thing we want to do is to be able to know which hosts ran a certain binary, and in our case the malware. There are two main ways we can accomplish this, which we'll call "Polling" and "Interrupting":

2.1 Polling

In this method, either the C2 server or the victim (Depending on our specific architecture) sends a packet to the other each time an interval of time we define passes, and it needs to respond with a certain known response. This way, we can make sure that the victim is still alive. Figure 2 shows this in more graphic detail:

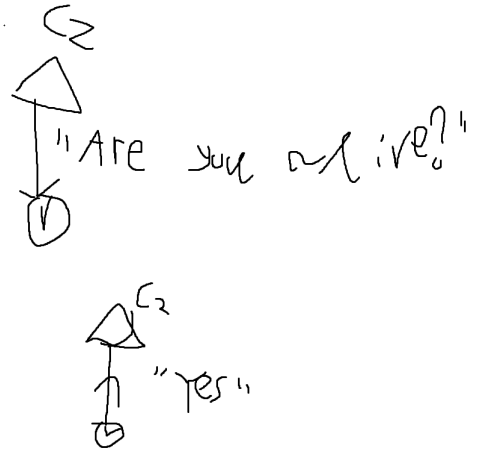

Figure 2: The process of polling a given victim

This process is repeated at regular intervals of time. If the victim doesn't respond after, say, 1 minute, we'll mark the victim as dead. Otherwise, it is known to be alive. The main downside of this method is that it's more noisy. For a network administrator, seeing a constant exchange of info between two hosts regularly is a very suspicious thing.

2.2 Interrupting

This method is simpler and less noisy than the previous method. Every time we ask the malware to do something, say give us a remote shell, we just ask the malware to respond with a given packet to tell the C2 server that it acknowledged the request. This figure shows it more clearly:

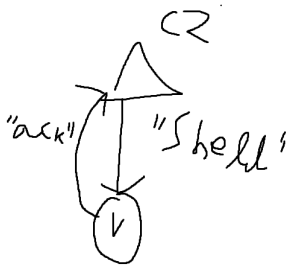

Figure 3: Interrupting