Take a deep breath: a Stealthy, Resilient and Cost-Effective Botnet Using Skype

Antonio Nappa - *Università degli studi di Milano*
Aristide Fattori - *Università degli studi di Milano*
Marco Balduzzi - *Eurecom, Sophia-Antipolis, France*
Matteo Dell'Amico - *Eurecom, Sophia-Antipolis, France*
Lorenzo Cavallaro - *Vrije Universiteit Amsterdam, The Nederlands*

Introduction

- Botnets are something that spreads along with "social software" (IRC, MSN, Skype, P2P clients, Facebook, Twitter).
- We observed the evolution of the botnet phenomenon and we individuate Skype as a possibile target to easily create a new botnet command and control channel.

Why Skype?

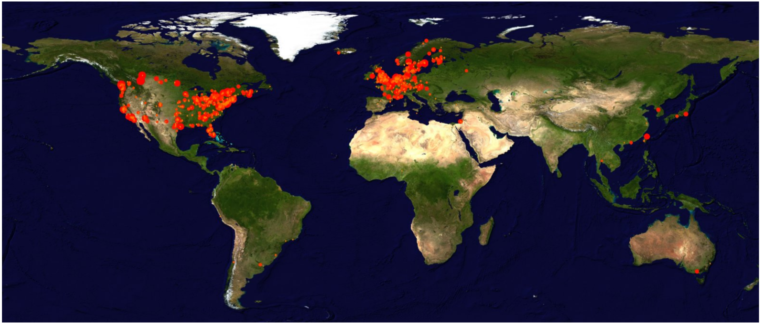
- We choose Skype because it is a widespread application, it has about 400 million of registered users and a daily presence of 50 million of users. Furthermore we choose it because it has a lot of appealing functionalities (bypass NAT and firewalls, encrypted communications).
- We wanted to proof that is cost-effective and fast to build a botnet with this application API in order to validate our model.

Advantages

Our solution is cost-effective because it is easy to deploy and has a lot of good functionalities (NAT and firewall passthrough, P2P structure)

- Botnet traffic indistiguishable from ordinary traffic
- No bottlenecks nor single point of failure
- Resiliency as the loss of one or many bots influences the infrastructure only slightly.
- Dynamic and transparent routing based on the Skype's Usernames.
- We take advantage of Skype's protection measures and P2P routing algorithms.

Supernodes

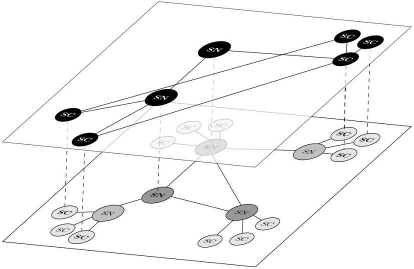


Botnet

We propose a novel botnet model that exploits an overlay network such as Skype to build a parasitic overlay.

Our Model

The parasitic overlay model is a botnet built on top of an instant messaging infrastructure using its features for non-standard operations.



Our model is generic and can be shaped on different applications that support instant messaging.

Features

- it is hard to set bots and regular Skype traffic apart
- the lack of hierarchical structure allows to use any controlled node as an entry point for the master
- the policy adopted for registering new nodes makes it cost-unattractive to obtain a comprhensive list of all the bots.

Communication protocol

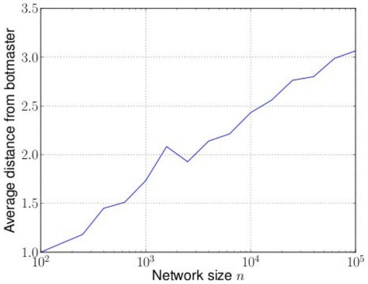
- To bootstrap each infected node sends a startup messages to its *Gate Nodes* embedded in the binary.
- The *Gate Nodes* are in contact with other nodes and the Master.
- The startup messages flows through the *Gate Nodes* and reach the Master.
- The Master answers to the infected node with a new set of neighbors.
- When the infected node has its new set of neighbors is able to communicate.

Communication protocol (2)

- In our botnet messages exchanged between bots and the master flow through the network as legitimate messages
- Usage of encryption to obtain unicast, multicast and broadcast communication
- Gnutella-like message passing procedure
- Ability to react in case of a total takeover of the *Gate Nodes*

Experiments

We evaluated our model through accurate simulations recreating different botnet magnitudes and connectivity states (alive neighbors per node).



The average distance between a node and the botmaster grows slowly with respect to the number of nodes in the botnet.

Experiments (2) - Proof of Concept

We created a small real-world scenario to verify our simulations results:

- PoC bot written in Python through Skype4Py libraries
- $\sim$ 40 hosts geographically distributed between France and Italy
- bootstrapping phase test, validation and measurements
- communication model test, validation and measurements

We observed that the real-world scenario is compliant with the simulated one.

Limitations

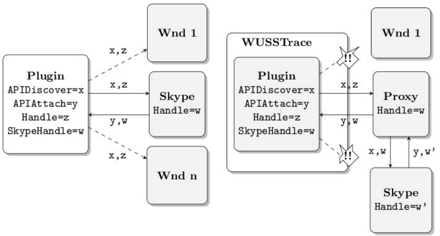
One important limitation of our Skype botnet is the possibility of an external attacker to perpetrate a replay attack.
This attack is done by repeatedly delivering announce messages to progressively obtain neighbor nodes lists during the bootstrap phase to obtain a map of the botnet.
One possible mitigation is to limit the number of neighbor nodes sent to new bots within a defined temporal window.

Discussion

The availability of the API that can interact with Skype with full privilege raises security issues.

Countermeasures

We developed a host-based countermeasure that intercepts the communications between the Skype API and a plugin, acting as a proxy. With this technique we are able to recognize every command issued by a plugin and we aim to find malicious command sequences. At the moment the rate of false positive is quite high. We are working on new heuristics to reduce this rate.



Questions