Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Botmaster Attribution in Large-Scale P2P Botnets

at the area of work ISS

Lars Leo Grätz

June 2019

1 Introduction

IT Security plays an important part in todays lifestyle. Not only is private data often digitaly stored by individuals, relevant businesses as well as infrastructure is also accessible via the Internet. This results in ambiguous attack vectors on the integrity, confidentiality and availability of these components, as well as the possibility of a machine being overtaken by an attacker. The execution of malware on a confiscated machine makes any attack that can be written in code possible. Botnets are a set of hacked machines, that execute the commands of a botmaster, the person in control. To do this the botmaster first has to infect devices that are connected to the internet. Once he has control over a number of bots, he is able to propagate commands that the machines will execute. This used to be done centralized via a command server, where the botmaster would leave a new piece of code, that all infected machines downloaded. Shutting down or "sinkholing" such a central server to get information about the botnet was relatively easy. Nowadays however, P2P propagation methods are far more common. In this distribution model, messages are propagated between infected machines and not via a centralized method. Concluding this, it is obvious that an army of confiscated bots that can arbitrarily be controlled without any trace of the attacker is really dangerous. This results in P2P botnet monitoring being an ever important task, especially to find out about propagation techniques as well as entry points of the botmaster. This thesis states how one could possibly attribute botmasters, focusing on the P2P botnet Sality. Different malware propagation methods are evaluated and crawlers were created to show possible information gathering mechanisms in P2P networks.

2 General P2P Botnet Architecture

In P2P networks, bots can be differentiated into superpeers, which are routable servers that can directly be contacted, and peers, which are not routable and thus have to poll information from superpeers [1].
In general each superpeer in a P2P botnet holds a list of neighbours (also superpeers), that can directly be contacted. This neighbourlist differs between bots, since it is dynamic and thus changes over time, depending on the accessibility of the neighbours. Each bot runs periodic membership maintenance (MM) in order to identify non responsible bots within its neighbourlist and also identify reliable ones. Non responsive bots are often discarded at a given point. This also means that a superpeer will try to gain new neigbhbours, once its neighbourlist reaches a low threshold of entries. This is done by contacting reliable neighbours and polling from their neighbourlist. [1].
It can ben stated, that in general P2P connected botnets are more resilient to simpler defense mechanism, since they do not have a singular point of failure.

2.1 Sality Dissection

As stated earlier, the main focus of this thesis lies on the botnet Sality, a polymorphic file infector for windows executables. Sality infects machines by concatenating the malicious payload to valid windows binaries. Then the entry point of the binary is obscured, such that it executes the malicious code, and afterwards jumps back to the original binary [2]. New malicious malware can easily be deployed at any time by substituting the appendix of the infected file. This included malware can then be used to carry out a number of tasks, such as shutting down services, deleting/encrypting files, sending spam, using the host for computational tasks etc.. The propagation of new malware through the Sality botnet is as follows: The malicious code is deployed to certain servers by the botmaster. He then proceeds to distribute a URL Pack throughout the network, a message of links to the servers that host the new version of the malicious code. This means, that the botmaster can use different IP Adresses each time, since they will be part of the propagated URL Pack. Each bot that receives one such pack downloads and executes the new malware [3]. This leads to the necessity of URL Packs having a unique sequence number, since a bot should not download outdated malware. When a new URL Pack is released, this number is simply incremented, such that different versions can be discriminated. This necessarily leads to the situation of different URL Pack versions being present in a snapshot of the botnet at times, when a new Pack has just been released by the botmaster.
Salitys superpeers typically hold a neighbourlist of up to 1000 entries, that additionally contains the LastOnline timestamp, a GoodCount, IP adress, Port and UID for each neighbour. The MM cycle is invoked every 40 minutes, which starts the following processes for each neighbour sequentially:

1. Probe the responsiveness using a probe message. On a successful response, the LastOnline timestamp is set to the current time and the GoodCount is incremented. If a timeout occurs or the bot is unresponsive, the GoodCount is decremented. With the probe message, the current sequence number is also delivered. Depending on the sequence number of the receiving bot, it will either ask for the whole URL pack, if its sequence number is lower or send back its own URL pack, if it is higher.

2. The superpeer status is tested. This is necessary for a bot to know if it is a superpeer and can propagate messages. When a bot is initialized it starts off with UID = 0, meaning its superpeer capabilities are unknown. If it has any other UID it is a superpeer. The process however is not relevant for this thesis and will be omitted for brevity, since the crawler can only trace superpeers.

3. If the size of the own neighbourlist is < 980 and the neighbour has a high GoodCount, it is also probed for a neighbour entry. In Sality each neighbour will respond with one randomly chosen entry, that has a high GoodCount.

After the cycle, a cleanup process takes place. Bots that have a GoodCount $<$ 30 are dropped from the list, if the size of the neighbourlist is at least 500. [1].

3 Requirements and related work

3.1 Requirements

The main focus of this thesis lies in evaluating different malware distribution techniques used in Sality and creating specialized crawlers that try to find a possible subset of superpeers that have a high chance of being connected to the botmaster. Regular monitoring techniques used to get an estimation of the botnet size as described in [1] can not be applied in this case, since the goal is not to find all members of the botnet, but rather: Given all members and connections, find the source of malware propagation. In order to narrow down the set of closely connected peers, the fact that new commands are propagated through the network in sequence, resulting in stepwise updates of individual bots, is used. Realizing this, the main lead for the crawlers is to identify bots that have a higher current version than others and traverse the network accordingly.

Functional Requirements:

1. Identification of botmaster strategies: The propagation statistics of URL Packs via different strategies are evaluated. Special emphasis lies on the difference between the botmaster pushing the message actively and letting the superpeers pool the new URL Pack.

2. Narrowing down botmaster entrypoints: Depending on the botmaster strategy and propagation statistics, the according crawlers are able to find a certain set of superpeers that is likely to be connected to the botmaster.

3. (Evaluating botmaster location: If the botmaster is part of the network, he should ultimately be found. Else, a possible set of outside adresses should be observed and evaluated.)

4. Genericity: The main algorithms and crawler methods work in different P2P botnets by implementing the specific communication protocols.

Nonfunctional Requirements:

1. Scalability: The speed of the crawler scales with the size of a botnet by adding more processing power.

2. Efficiency: The crawler avoids unnecessary overhead and only exchanges messages that are needed.

3. Avoiding detection: The crawler works around popular botnet defense mechanism that detect crawlers.

4 Botmaster Strategies

This is the main focus for the first part of the thesis. The botmaster can either be part of the network and increment his own sequence number to let the other superpeers poll the new version, or push the new URL Pack to a list of neighbours himself. Furthermore it can be differentiated, if the botmaster uses the whole URL Pack protocol, or just pushes the packs directly.

5 System Design

Since the evaluation of real existing botnets is hard because of various reasons, such as other crawlers/sensors spoiling the collected data, the churn rate of the network (the rate at which peers join/leave the network) and last but not least the nondeterministic, uncontrollable environment that is hard to evaluate discretely, a simulated environment is used.

The following notation is used in the thesis:

- $G_B = (V, E)$: the graph representation of the botnet.
- $V_S \subset V$: The set of superpeers.
- $N_v,\ v \in V$ The neighbourlist of a bot. This list contains the bots he is directly connected to.
- $e \in E = (u, v),\ u, v \in V$: a directed connection between two superpeers. This means that v is in the neighbourllist of u.
- seq_v: The URL Pack sequence number of superpeer $v \in V_S$.

For the testing phase of the crawler, a simulation environment in OMNeT++ (Objective Modular Network Testbed in C++) has been created. This environment features a realistic implementation of Salitys protocols as well as Superpeer behaviour. To add even more realism, the structure of the simulated network can be imported from real snapshots of the existing botnet via a Parser (written in Python) that takes a GraphML file and returns a NED file used in OMNeT++. Regular peers are of no interest in the simulation, since they can not propagate URL packs. The following different crawler versions are used:

- Crawler V1: Crawling based on package sequence numbers: In this approach, the crawler constantly maintains a set of eligible superpeers V_E that possibly are connected to the botmaster and also saves the highest found sequence number seq_{max}. V_E is initialized as V_s. It now works iteratively in cycles. For each bot $u \in V_E$ it pulls seq_u. Then the maximum of seq_u over all bots seq_{u_max} is determined. seq_{max} is set to seq_{u_max}. All bots $u \in V_E$ are iterated and discarded, if $seq_u < seq_{max}$. Afterwards the cycle repeats.

The algorithms are implemented in two different botnet architectures. First off, a static snapshot of a botnet is used. This means, that node churn is completely ignored, and the neighbourlists also stay the same. Secondly node churn is enabled, leading to nondeterministic behaviour of the crawler, but a more realistic representation of the actual botnet. This leads to the necessity of the algorithms being able to work with dynamic neighbourlists.

In order to evaluate the data, it is written to log files during the simulation process. These log files are used for statistical analysis. This provides the groundwork for further processing of the information in order to retrieve insights to network stats and propagation statistics.

6 Evaluation

Different metrics are established to measure the success of the crawler:

1. Size of subset $V_E \subset V_S$ of superpeers potentially connected to the botmaster. The smaller this size, the better the crawler.
2. Average steps of a superpeer p for $p \in V_E$ to the initial superpeer that received the package. The higher this metric, the worse the crawler performed.
3. Number/percentage of superpeers p for $p \in V_S, p \notin V_E$ that are closer to the initial source as stated in 2.. This is the amount of superpeers that have not been found, but are potentially connected to the botmaster.

Thema: Using Sality URL Packs to narrow down the superpeers directly in contact with the botmaster
Bearbeiter: Lars Leo Grätz
Datum: June 7, 2019

References

[1] S. Karuppayah, *Advanced Monitoring in P2P Botnets. A Dual Perspective.* Springer, 2018.

[2] Wikipedia, "Sality."

[3] N. Falliere, *Sality: Story of a Peer-to-Peer Viral Network.* Symantec Corporation, 2011.