OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

MASTERTHESIS

Valentin Brandl

Collaborative Crawling of Fully Distributed Botnets

February 27, 2022

Faculty:	Informatik und Mathematik
Study Programme:	Master Informatik
Supervisor:	Prof. Dr. Christoph Skornia
Secondary Supervisor:	Prof. Dr. Thomas Waas

TODO: abstract

Contents

1 Introduction

The internet has become an irreplaceable part of our day to day lives. We are always connected via numerous “smart” and internet of things (IoT) devices. We use the internet to communicate, shop, handle financial transactions and much more. Many personal and professional workflows are so dependent on the internet, that they won’t work when being offline.

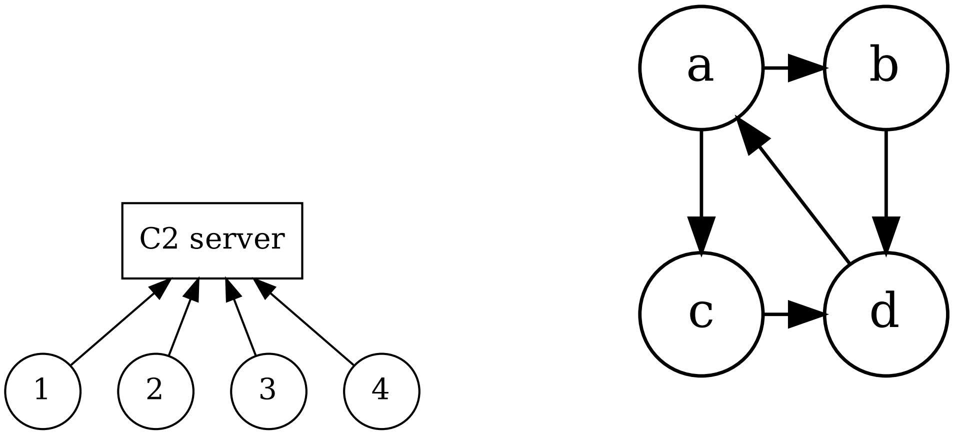
1.1 Motivation

The number of connected IoT devices is around 10 billion in 2021 and estimated to be constantly growing over the next years up to 25 billion in 2030 [7]. Many of these devices run on outdated software, don’t receive any updates and don’t follow general security best practices. While in 2016 only 77% of German households had a broadband connection with a bandwidth of 50 Mbit/s or more, in 2020 it were already 95% with more than 50 Mbit/s and 59% with at least 1000 Mbit/s [3]. This makes them an attractive target for botmasters since they are easy to infect, always online, behind internet connections that are getting faster and faster, and due to their nature as small devices, often without any direct user interaction, an infection can go unnoticed for a long time. In recent years, IoT botnets have been responsible for some of the biggest distributed denial of service (DDoS) attacks ever recorded, creating up to 1 Tbit/s of traffic [5].

A botnet describes a network of connected computers with some way to control the infected systems. In classic botnets, there are one or more central coordinating hosts called command and control (C2) servers. These C2 servers could use anything from internet relay chat (IRC) over hypertext transfer protocol (HTTP) to Twitter as communication channel with the infected systems. The infected systems can be abused for a number of things, *e.g.* DDoS attacks, stealing data from victims, as proxies to hide the attackers identity, send spam emails…

Analyzing and shutting down a centralized botnet is comparatively easily since every bot knows the IP address, domain name, Twitter handle or IRC channel the C2 servers are using.

A targeted operation with help from law enforcement, hosting providers, domain registrars and platform providers could shut down or take over the operation by changing how requests are rooted or simply shutting down the controlling servers/accounts.



(a) Topology of a C2 controlled botnet (b) Topology of a peer-to-peer (P2P) botnet

Figure 1: Communication paths in different types of botnets

A number of botnet operations were shut down like this and as the defenders upped their game, so did attackers — the idea of peer-to-peer (P2P) botnets came up. The idea is to build a decentralized network without single points of failure where the C2 servers are as shown in Figure 1b. In a P2P botnet, each node in the network knows a number of it's neighbours and connects to those, each of these neighbours has a list of neighbours on his own, and so on.

This lack of a single point of failure (SPOF) makes P2P botnets more resilient to take-down attempts since the communication is not stopped and botmasters can easily rejoin the network and send commands.

Formally, a P2P botnet can be modeled as a digraph

$$G = (V, E)$$

With the set of vertices V describing the bots in the network and the set of edges E describing the “is neighbour of” relationships between bots. For a vertex $v \in V$, the in and out degree $\deg^+$ and $\deg^-$ describe how many bots know v or are known by v respectively.

$$\deg^+(v) = |\{u \in V \mid (u,v) \in E\}|$$
$$\deg^-(v) = |\{u \in V \mid (v,u) \in E\}|$$

For a vertex $v \in V$, the in degree $\deg^+(v) = |\{u \in V \mid (u,v) \in E\}|$ and out degree $\deg^-(v) = |\{u \in V \mid (v,u) \in E\}|$ describe how many bots know v and how many nodes v knows respectively.

The damage produced by botnets has been constantly growing and there are many researchers and law enforcement agencies trying to shut down these operations. The monetary value of these botnets directly correlates with the amount of effort, botmasters are willing to put into implementing defense mechanisms against take-down attempts. Some of these countermeasures include deterrence, which limits the amount of allowed bots per IP address or subnet to 1; blacklisting, where known crawlers and sensors are blocked from communicating with other bots in the network (mostly IP based); disinformation, when fake bots are placed in the neighbourhood lists, which invalidates the data collected by crawlers; and active retaliation like DDoS attacks against sensors or crawlers [1].

1.2 Detection Techniques for P2P Botnets

There are two distinct methods to map and get an overview of the network topology of a P2P botnet:

1.2.1 Passive Detection

For passive detection, traffic flows are analyzed in large amounts of collected network traffic (*e.g.* from internet service providers (ISPs)). This has some advantages in that it is not possible for botmasters to detect or prevent data collection of that kind but it is not trivial to distinguish valid P2P application traffic (*e.g.* BitTorrent, Skype, cryptocurrencies, ...) from P2P bots. Zhang et al. propose a system of statistical analysis to solve some of these problems in [9]. Also getting access to the required datasets might not be possible for everyone.

- Large scale network analysis (hard to differentiate from legitimate P2P traffic (*e.g.* BitTorrent), hard to get data, knowledge of some known bots required) [9]
- Heuristics: Same traffic patterns, same malicious behaviour

1.2.2 Active Detection

In this case, a subset of the botnet protocol are reimplemented to place pseudo-bots or sensors in the network, which will only communicate with other nodes but won't accept or execute commands to perform malicious actions. The difference in behaviour from the reference implementation and conspicuous graph properties (*e.g.* high $\deg^+$ vs. low $\deg^-$) of these sensors allows botmasters to detect and block the sensor nodes.

There are three subtypes auf active detection:

1. Crawlers: recursively ask known bots for their neighbourhood lists
2. Sensors: implement a subset of the botnet protocol and become part of the network without performing malicious actions
3. Hybrid of crawlers and sensors

1.3 Detection Criteria

- P2P online time vs host online time
- neighbourhood lists
- no/few domain name system (DNS) lookups; instead direct lookups from routing tables

2 Methodology

The implementation of the concepts of this work will be done as part of Botnet Monitoring System (BMS)[1], a monitoring platform for P2P botnets described by Böck et al. in [4]. BMS uses a hybrid active approach of crawlers and sensors (reimplementations of the P2P protocol of a botnet, that won't perform malicious actions) to collect live data from active botnets.

In an earlier project, I implemented different node ranking algorithms (among others "PageRank" [8]) to detect sensors and crawlers in a botnet, as described in "SensorBuster". Both ranking algorithms use the $\deg^+$ and $\deg^-$ to weight the nodes. Another way to enumerate candidates for sensors in a P2P botnet is to find weakly connected components (WCCs) in the graph. Sensors will have few to none outgoing edges, since they don't participate actively in the botnet.

The goal of this work is to complicate detection mechanisms like this for botmasters, by centralizing the coordination of the system's crawlers and sensors, thereby reducing the node's rank for specific graph metrics. The changes should allow the current sensors to use the new abstraction with as few changes as possible to the existing code.

The final result should be as general as possible and not depend on any botnet's specific behaviour but it assumes, that every P2P botnet has some kind of "getNeighbourList" method in the protocol, that allows other peers to request a list of active nodes to connect to.

In the current implementation, each sensor will itself visit and monitor each new node it finds. The idea for this work is to report newfound nodes back to the BMS backend first, where the graph of the known network is created and a sensor is selected, so that the specific ranking algorithm doesn't calculate to a suspiciously high or low value. That sensor will be responsible to monitor the new node.

If it is not possible, to select a specific sensor so that the monitoring activity stays inconspicuous, the coordinator can do a complete shuffle of all nodes between the sensors to restore the wanted graph properties or warn if more sensors are required to stay undetected.

---

[1] `https://github.com/Telecooperation/BMS`

The improved sensor system should allow new sensors to register themselves and their capabilities (*e.g.* bandwidth, geolocation,), so the amount of work can be scaled accordingly between hosts. Further work might even consider autoscaling the monitoring activity using some kind of cloudcomputing provider.

To validate the result, the old sensor implementation will be compared to the new system using different graph metrics.

If time allows, Botnet Simulation Framework (BSF)[2] will be used to simulate a botnet place sensors in the simulated network and measure the improvement achieved by the coordinated monitoring effort.

As a proof of concept, the coordinated monitoring approach will be implemented and deployed in the (Sality, Mirai, ...)? botnet.

2.1 Protocol Primitives

The coordination protocol must allow the following operations:

2.1.1 Sensor to Backend

- `registerSensor(capabilities)`: Register new sensor with capabilities (which botnet, available bandwidth, . . .). This is called periodically and used to determine which crawler is still active, when splitting the workload.
- `unreachable(targets)`:
- `requestTasks() []PeerTask`: Receive a batch of crawl tasks from the coordinator. The tasks consist of the target peer, if the crawler should start or stop the operation, when it should start and stop monitoring and the frequency.

```
type Peer struct {
  BotID string
  IP    string
  Port  uint16
```

---

[2] `https://github.com/tklab-tud/BSF`

```
}
type PeerTask struct {
  Peer         Peer
  StartAt      *Time
  StopAt       *Time
  Frequency    uint
  StopCrawling bool
}
```

2.1.2 Backend to Sensor

3 Coordination Strategies

3.1 Reduction of Request Frequency

The GameOver Zeus botnet deployed a blacklisting mechanism, where crawlers are blocked based in their request frequency [2]. In a single crawler approach, the crawler frequency has to be limited to prevent being hitting the request limit. Using collaborative crawlers, an arbitrarily fast frequency can be achieved without being blacklisted. With $L \in \mathbb{N}$ being the frequency limit at which a crawler will be blacklisted, $F \in \mathbb{N}$ being the crawl frequency that should be achieved. The amount of crawlers C required to achieve the frequency F without being blacklisted and the offset O between crawlers are defined as

$$
\begin{aligned}
C &= \left\lceil \frac{F}{L} \right\rceil \\
O &= \frac{1 \text{req}}{F}
\end{aligned}
$$

Taking advantage of the `StartAt` field from the `PeerTask` returned by the `requestTasks` primitive above, the crawlers can be scheduled offset by O at a frequency L to ensure, the overall requests to each peer are evenly distributed over time.

Given a limit $L = 5\text{req}/100\text{s}$, crawling a botnet at $F = 20\text{req}/100\text{s}$ requires $C = \left\lceil \frac{20\text{req}/100\text{s}}{5\text{req}/100\text{s}} \right\rceil = 4$ crawlers. Those crawlers must be scheduled $O = \frac{1\text{req}}{20\text{req}/100\text{s}} = 5\text{s}$ apart at a frequency of L for an even request distribution.

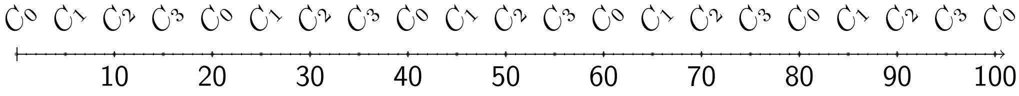
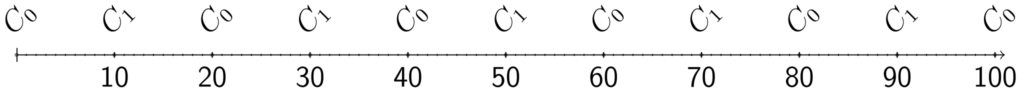


Figure 2: Timeline of crawler events as seen from a peer

As can be seen in Figure 2, each crawler C_0 to C_3 performs only 5 req/100s while overall achieving $20\text{req}/100\text{s}$.

Vice versa given an amount of crawlers C and a request limit L, the effective frequency F can be maximized to $F = C \times L$ without hitting the limit L and being blocked.

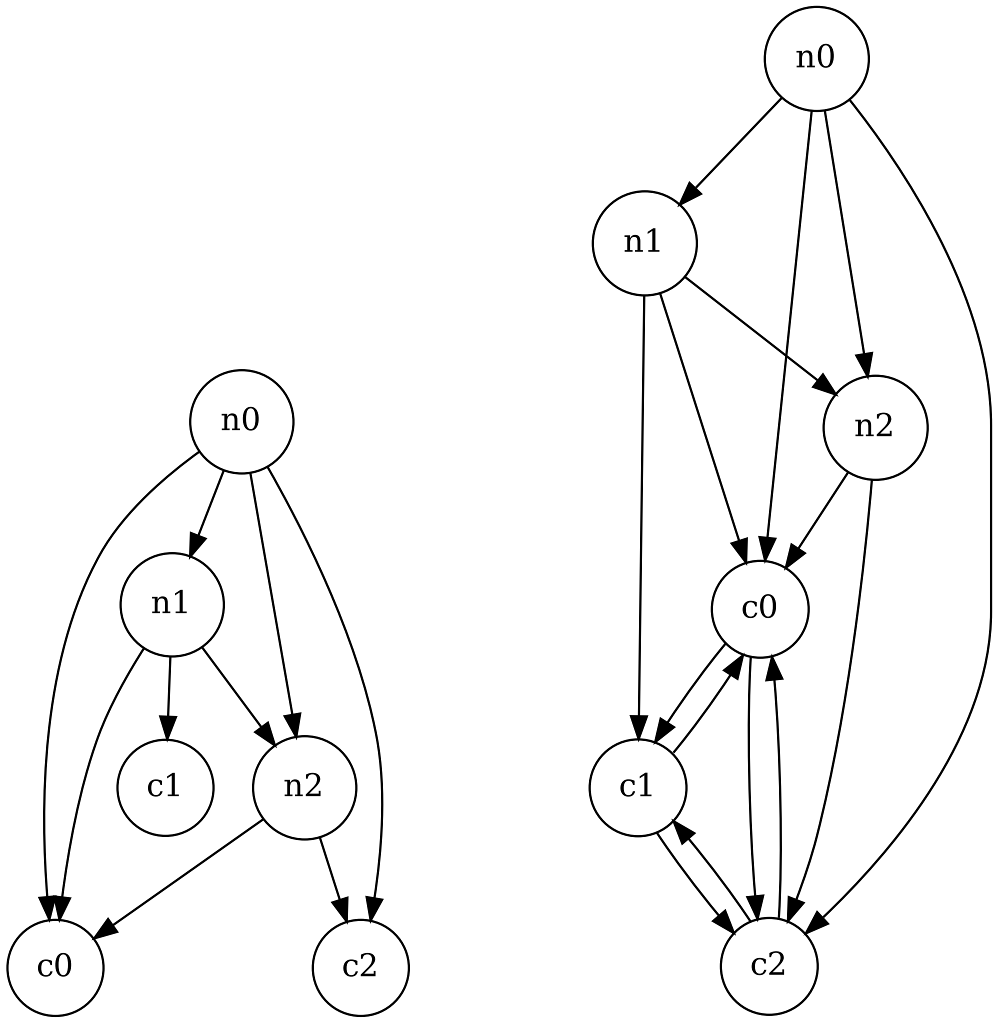
Using the example from above with $L = 5\text{req}/100\text{s}$ but now only two crawlers $C = 2$, it is still possible to achieve an effective frequency of $F = 2 \times 5\text{req}/100\text{s} = 10\text{req}/100\text{s}$ and $O = \frac{1\text{req}}{10\text{req}/100\text{s}} = 10\text{s}$:



While the effective frequency of the whole system is halved compared to Figure 2, it is still possible to double the frequency over the limit.

3.2 Working Against Suspicious Graph Metrics

“SensorBuster: On Identifying Sensor Nodes in P2P Botnets” describes different graph metrics to find sensors in P2P botnets. One of those, “SensorBuster” uses WCCs since crawlers don’t have any edges back to the main network in the graph. It would be possible to implement the crawlers so they return other crawlers in their peer list responses but this would still produce a disconnected component and as long as this component is smaller than the main network, it is still easily detectable since there is no path from the crawler component back to the main network.



(a) WCCs for independent crawlers (b) WCCs for collaborated crawlers

Figure 3: Differences in graph metrics

Node	$\deg_a^+$	$\deg_a^-$	$\deg_b^+$	$\deg_b^-$
n0	0	4	0	4
n1	1	3	1	3
n2	2	2	2	2
c0	3	0	5	2
c1	1	0	3	2
c2	2	0	4	2

References

[1] Dennis Andriesse, Christian Rossow, and Herbert Bos. "Reliable Recon in Adversarial Peer-to-Peer Botnets". In: *Proceedings of the 2015 Internet Measurement Conference*. IMC '15: Internet Measurement Conference. Tokyo Japan: ACM, Oct. 28, 2015, pp. 129–140. ISBN: 978-1-4503-3848-6. DOI: `10.1145/2815675.2815682`. URL: `https://dl.acm.org/doi/10.1145/2815675.2815682` (visited on 11/16/2021).

[2] Dennis Andriesse et al. "Highly Resilient Peer-to-Peer Botnets Are Here: An Analysis of Gameover Zeus". In: *2013 8th International Conference on Malicious and Unwanted Software: "The Americas" (MALWARE)*. 2013 8th International Conference on Malicious and Unwanted Software: "The Americas" (MALWARE). Fajardo, PR, USA: IEEE, Oct. 2013, pp. 116–123. ISBN: 978-1-4799-2534-6 978-1-4799-2535-3. DOI: `10.1109/MALWARE.2013.6703693`. URL: `https://ieeexplore.ieee.org/document/6703693/` (visited on 02/27/2022).

[3] *Availability of broadband internet to households in Germany from 2017 to 2020, by bandwidth class*. Statista Inc. Aug. 16, 2021. URL: `https://www.statista.com/statistics/460180/broadband-availability-by-bandwidth-class-germany/` (visited on 11/11/2021), archived at `https://web.archive.org/web/20210309010747/https://www.statista.com/statistics/460180/broadband-availability-by-bandwidth-class-germany/` on Mar. 9, 2021.

[4] Leon Böck et al. "Poster: Challenges of Accurately Measuring Churn in P2P Botnets". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. CCS '19: 2019 ACM SIGSAC Conference on Computer and Communications Security. London United Kingdom: ACM, Nov. 6, 2019, pp. 2661–2663. ISBN: 978-1-4503-6747-9. DOI: `10.1145/3319535.3363281`. URL: `https://dl.acm.org/doi/10.1145/3319535.3363281` (visited on 11/12/2021).

[5] Dan Goodin. *Brace yourselves — source code powering potent IoT DDoSes just went public*. Ars Technica. Oct. 2, 2016. URL: `https://arstechnica.com/information-technology/2016/10/brace-yourselves-source-code-powering-potent-iot-ddoses-just-went-public/` (visited on 11/11/2021), archived at `https://web.archive.org/web/20211022032617/https://arstechnica.com/information-technology/2016/10/brace-yourselves-source-code-powering-potent-iot-ddoses-just-went-public/` on Oct. 22, 2021.

[6] Shankar Karuppayah et al. “SensorBuster: On Identifying Sensor Nodes in P2P Botnets”. In: *Proceedings of the 12th International Conference on Availability, Reliability and Security*. ARES ’17. New York, NY, USA: Association for Computing Machinery, Aug. 29, 2017, pp. 1–6. ISBN: 978-1-4503-5257-4. DOI: `10.1145/3098954.3098991`. URL: `https://doi.org/10.1145/3098954.3098991` (visited on 03/23/2021).

[7] *Number of Internet of Things (IoT) Connected Devices Worldwide from 2019 to 2030*. Statista Inc. Dec. 2020. URL: `https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/` (visited on 11/11/2021), archived at `https://web.archive.org/web/20211025185804/https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/` on Oct. 25, 2021.

[8] Lawrence Page et al. *The PageRank Citation Ranking: Bringing Order to the Web*. Jan. 29, 1998. URL: `http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf` (visited on 11/30/2021).

[9] Junjie Zhang et al. “Building a Scalable System for Stealthy P2P-Botnet Detection”. In: *IEEE Transactions on Information Forensics and Security* 9.1 (Jan. 2014), pp. 27–38. ISSN: 1556-6013, 1556-6021. DOI: `10.1109/TIFS.2013.2290197`. URL: `http://ieeexplore.ieee.org/document/6661360/` (visited on 11/09/2021).

List of Figures

Acronyms

BMS Botnet Monitoring System

BSF Botnet Simulation Framework

C2 command and control

DDoS distributed denial of service

DNS domain name system

HTTP hypertext transfer protocol

IoT internet of things

IRC internet relay chat

ISP internet service provider

P2P peer-to-peer

SPOF single point of failure

WCC weakly connected component

Erklärung

1. Mir ist bekannt, dass dieses Exemplar der Masterthesis als Prüfungsleistung in das Eigentum der Ostbayerischen Technischen Hochschule Regensburg übergeht.
2. Ich erkläre hiermit, dass ich diese Masterthesis selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Ort, Datum und Unterschrift

Presented by:	Valentin Brandl
Student ID:	3220018
Study Programme:	Master Informatik
Supervisor:	Prof. Dr. Christoph Skornia
Secondary Supervisor:	Prof. Dr. Thomas Waas