



Proof of concept; a worm botnet.

Abigail ECCLESTON, B00140442

Jérémy PRIMARD, B00168148

Tytus KOPERA, B00140074

**School of Informatics and Cybersecurity, Faculty of Computing, Digital & Data,
Technological University Dublin**

**Submitted to Technological University Dublin in partial fulfillment of the requirements for
the degree of**

Bachelor of Science in Computing in Digital Forensics & Cyber Security

Supervisor:

Mark Lane

May 2024



Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Degree of **Bachelor of Science in Computing in Digital Forensics & Cyber Security** in Technological University Dublin, is entirely my own work except where otherwise stated, and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfilment of the requirements of that stated above.

Dated: 04/15/2023

Abigail ECCLESTON

Jérémy PRIMARD

Tytus KOPERA

Abstract

Abstract

Whatever one host can do by itself, one thousand hosts can do it better together. Malicious actors spread their networks of zombie hosts to grant them the power of many machines. They pose a challenge to law enforcement, companies and targeted individuals given their power to cause damage as well as resilience from being disrupted.

We decided to learn about the intricacies of this threat by creating our own botnet, written in Rust. The aim was to delve into the low level world of programming networking functionality, hence why Rust was used. We used the Rust library documentation and Rust's compiler to learn the language along the way and getting acquainted with how a networking program should be written in a low level. This botnet utilizes the ShellShock ie. CVE-2014-6271 on hosts running an outdated version of Apache. Communication is encrypted with TLS thanks to OpenSSL and Rustls. By design, only 1 to 1 parent-child node relations are possible, owing to the targeting system implemented.

The exploit itself was just as important as the botnet itself. Without this, the hosts could never have been infected by its parent server, and so would never have joined the network of zombie machines.

Despite the age of the exploit, it is very likely that many machines are running this outdated version of Apache, meaning that our malware could potentially ensnare multiple hosts, adding them to our botnet.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem.

Keywords: Botnet, malware, virus, rust, TODO.

Contents

Abstract	1
I Introduction	3
I.1 Definitions	3
I.2 Origins of the project	3
I.3 Preliminary research over an existing botnet and an existing worm that appears in the last 25 years.	3
II Our implementation and design	5
II.1 Things necessary for the project we had to either find or makes.	5
II.2 design explanations about the botnet.	6
II.3 lab setup.	6
II.4 Botnet communication.	7
III Part 2	9
III.1 Something part	9
III.2 Something part	9
III.3 Something part	9
IV Part 3	10
IV.1 Something part	10
IV.2 Something part	10
IV.3 Something part	10
Conclusion	11
Glossary	12
References	13

I. Introduction

1.1. Definitions

In a world where more and more things rely on computing, it is necessary to know how virus works, especially some of the worse ones. Worms, a type of virus that can self replicate in order to infect more and more. It is usually used to launch some ransomwares and other denial of service attacks. Because of its self replicated nature, once launched, it doesn't need a head to continue spreading if it's coded to do that. It also has the good ability, depending on what its purpose is, to conceal the pirate. Botnets, another type of virus that takes control of multiple machines in order to do evil things, like Distributed Denial of Services (DDoS) attacks, or win more power in order to either mine cryptomoney, or crack passwords are another.

1.2. Origins of the project

This is where the idea of the project started. Because one of the huge flaws of botnets is that they need a head to work, we think of this idea. What if we can allude the ability to conceal the perpetrator of the attack from the worm, and the ability to control bots of a botnet? More precisely, our idea is to make a botnet that spreads like a worm, so that only a very little quantity of machines actually knows the address of the head of the botnet. We remind here that our project is simply a proof of concept, and so, it's only meant to be shown in the final demo of the project. Several parts of it, if not all of them, would need real improvements, in order to overcome some of its limits. Again, we remind that this project isn't meant to show the virus of the future, but only to show, and give a proof of concept, of a type of botnet where there isn't enough research on it. Indeed, the closest thing we could find to our idea, was Peer-to-peer botnet, that's similar to what we want to do, but it wasn't enough. (<https://ieeexplore.ieee.org/abstract/document/5684002/>) (<https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=5de6987574e48076fa5f024f347d44c77a6fa080>) We can also add to it that we were interested by the field

1.3. Preliminary research over an existing botnet and an existing worm that appears in the last 25 years.

Before beginning to talk about the implementation of our botnet, let's find three examples of pretty well known worms and botnets, to get an idea of what already exists.

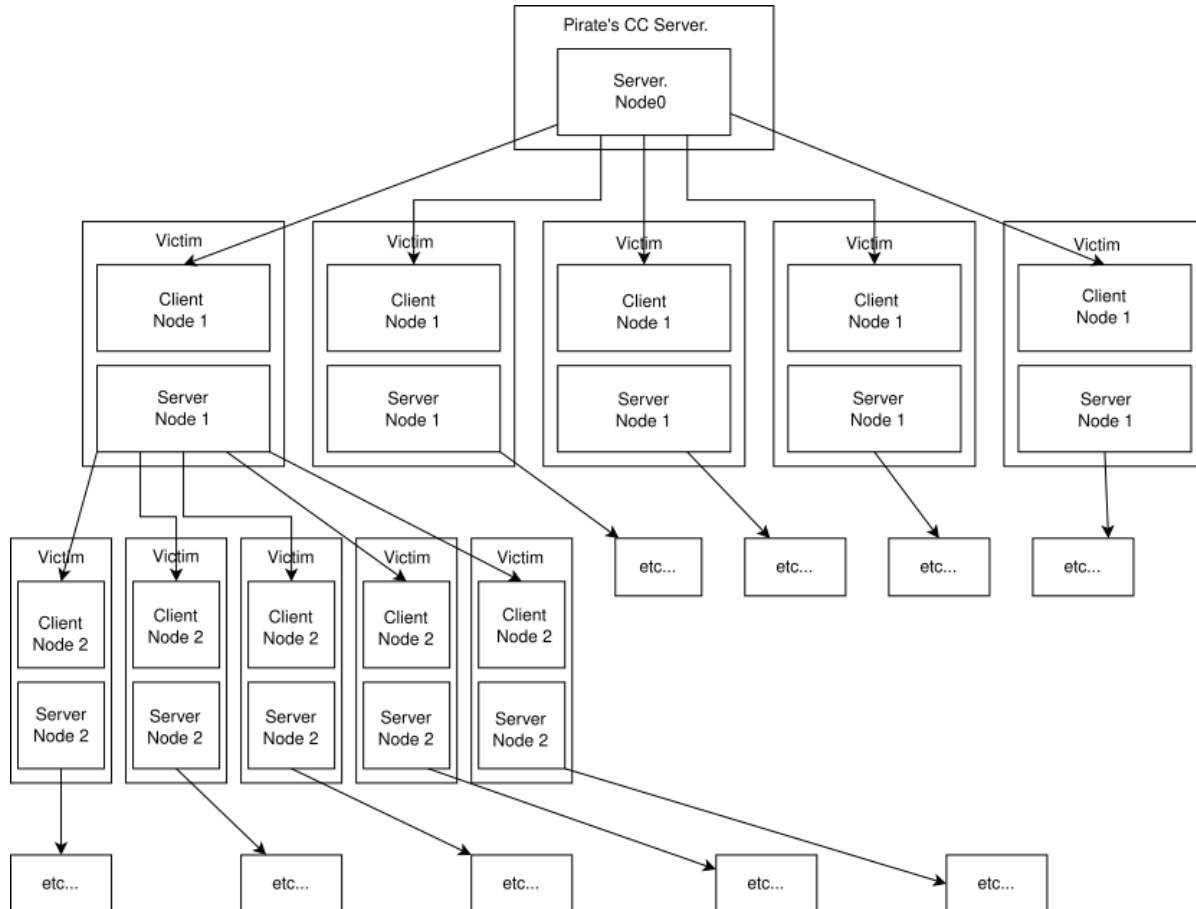
The first worm that we found was WannaCry. (https://en.wikipedia.org/wiki/WannaCry_ransomware_attack) It's a bot which appeared in 2017 and targeted only Microsoft Windows software. It was spread using a critical vulnerability of the SMB server that is on every Windows machine. The way it works is the same way as most worms, that means that if a machine is vulnerable, the worm will drop and execute a file, with the goal to first self-replicate the worm, install a ransomware, and target other machines. (<https://www.csoonline.com/article/563017/wannacry-explained-a-perfect-ransomware-storm.html>)

The other example we chose, is the open source, Mirai botnet. (<https://en.wikipedia.org/wiki/>

Mirai_(malware)) It's a botnet spreading over ports 23 and 2323, in order to make new bots. It's way of working is simple. It tries to bruteforce some linux iot devices, using the default built-in credential, assuming that nobody changed it. After that, the bot awaits it's orders. Usually, it was to DOS a DNS server or a big cloud company.

Another botnet example, is the so called Hajime botnet ([https://en.wikipedia.org/wiki/Hajime_\(malware\)](https://en.wikipedia.org/wiki/Hajime_(malware))) which goal was simply to protect device against the mirai botnet, by closing telnet ports. This is a really well know botnet, using peer to peer capabilities, to hide it's owner address, and thanks to that, makes it a lot more difficult to be blocked by ISP to takes down the botnet. (https://thehackernews.com/2017/04/vigilante-hacker-iot-botnet_26.html)

II. Our implementation and design



II.1. Things necessary for the project we had to either find or make.

Our project had different parts.

- Make a sandboxing environment, to test it, while being sure that it can't escape. We used qemu and unshare to make an environment that couldn't reach the external network. In the end, it wasn't really useful to make it, because we decided that for the demo, the victims' IP should be known at launch time, so that we won't see unexpected things in the final demo.
- Find an exploit to use. It had to be a remote code execution, easy to setup, and easy to exploit, because it's not in the purpose of the project to find a new vulnerability, make an exploit or another crazy thing like that. We spent hours looking for the perfect exploit, and in the end, we found somebody else's project on github, that was doing exactly what we were looking for. (<https://github.com/opsxcq/exploit-CVE-2014-6271>) Thanks to opsxcq's work, we had an easy vulnerability to exploit. We chose shellshock, CVE-2014-6271. We found it in opsxcq

repository, a vulnerable docker image to shellshock CVE-2014-6271, and an easy command to exploit it. The other two interesting vulnerabilities that we found in his repos, but that in the end we chose to not use, were CVE-2016-10033 and CVE-2017-7494. There were both unadapted to our needs because of how difficult they would make us write the exploit. We also found Log4J, but in the end decided that we would better to abandon it, and try an easier vulnerability to exploit.

- Demonstrate the botnet. Like all botnet run things, we had to find a way to prove that the botnet has really taken control of several machines. For that, we made one more VM, that we called vulnerableDOS. We simply run Wireshark into it, and once the botnet has taken control of the bots, it will make them ping our machine. At first, we wanted to make our botnet run an actual DDOS attack on the vulnerableDOS machine, but because it's not in the purpose of the project to make it run a complex attack, our goal is just to prove that we have indeed taken control of the vulnerable machines, we chose to do only ping.

Obviously, we also had to make the actual botnet.

II.2. design explanations about the botnet.

Design and make the botnet was the real challenge. This is what we chose to do. The architecture of our botnet will be a Server, that's gonna do three things. First, run the exploit on the victim. Second, wait for a response from the victim, to run a Command and Control server (CC), to deliver its order. The first order in the demo will always be to replicate the server on the victim, and the second one to ping the vulnerableDOS VM. Third, the server will run a file sharing server (fs), so that everytime the CC server asks the victim to download a file, the file can be downloaded.

On the CC and the fs server, we implement a TLS connection, using rustls. The certificates used are self signed certificates, and everytime the victim self-replicates the server, it will use the command openssl to generate the certificates. We were really lucky that on the vulnerable docker image that we found, there was openssl 1.0.1f installed. Maybe if the docker image was made one or two years earlier, there wouldn't be openssl installed, and we would have had to abandon the idea to use encrypted connections. We were really lucky.

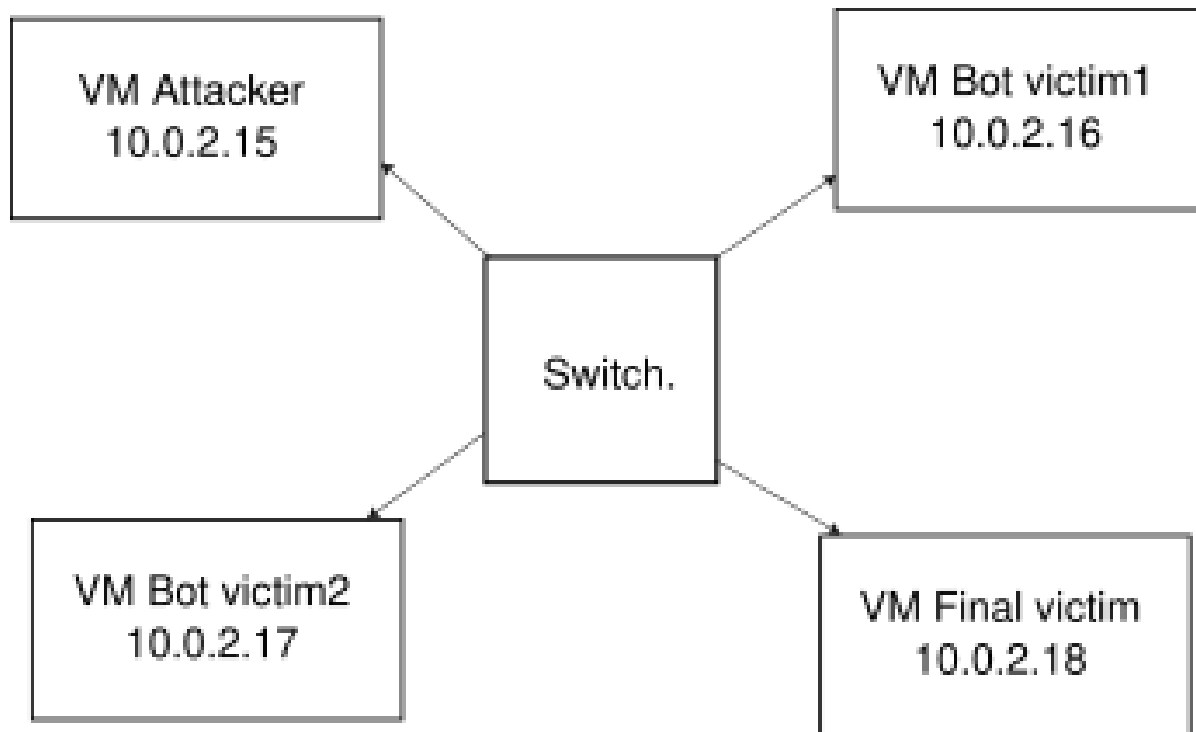
The architecture of our botnet will also be composed of a client. Its goal will be simply to connect to the server, get its order, and execute it whatever it is.

The server will read in a conf folder its TLS certificates, and the orders file that it will transmit to the victim. It will also share some scripts in a www folder, like the list of IPs that the next victim has to attack, a config file to easily generate new certificates, the exploit to run in order to run the client on the victim, and an installation script, in order to permit the client to once it has downloaded all the files it needed, to install them correctly, and self replicate the server.

II.3. lab setup.

To run the demo, we first made a base-VM, with runit-dinit on it, and only basic packages. Our goal was to know about everything installed, so that we don't lose time looking for where a bug comes from, when it's for a default package. Then, we made 1 VM to be the head of our botnet, 2 vulnerable VMs, so that when the botnet attacks them, they can become bots, and a last one, named

vulnerableDOS, with wireshark installed on it.



Our choice to test the botnet was to make all bots to ping the vulnerableDOS vm. That means that our botnet will work if we manage to make bot 1 and 2 to ping vulnerableDOS vm.

II.4. Botnet communication.

This is the normal fonctionnement of the botnet. Step one. The server find an potential target. then, it runs a BASH script, called exploit.sh, which exploit the target. That means, exploit the vulnerability, in order to make the victim download and execute the client. In our case, the client is installed and ran in '/tmp/botnet/'.

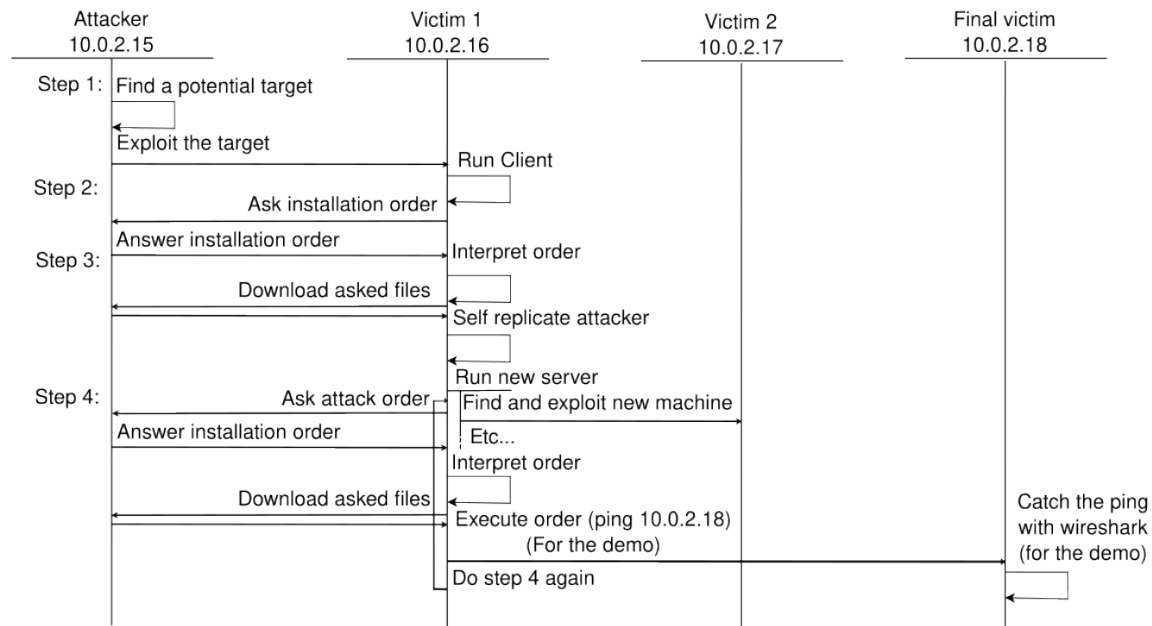
Step two. The client ask the server it's installed order, called order1. The server answer, and the client save it in '/tmp/botnet/conf/'.

Step three. The client read the received order, and every time it reach a line begining by 'download', it ask the filesharing server to give it, and put it in '/tmp/botnet/downloaded/'. Everytime the client reach a line begining by execute, it's execute the file.

Normally, the only executed file in the installation phase, it's the script 'installation.sh'. This script will copy all the downloaded files from '/tmp/botnet/downloaded' to their rightheous folders. It will then execute the server. The new server will look for other victims, and run again it's CC server and its fs server.

Step four. Finally, once the client had run another server, it will ask for it's attack order from the CC server, which is called order2. Typically, an order2 file will ask the client to download and execute a BASH script. This BASH script will run the attack. In the case of the demo, it will only

run 'ping 10.0.2.18' to prove that it has made a bot.



What the flowchart doesn't say, is that every time it receives an order, it will save it, and make it accessible to the server so that it can relate to the next node the orders. Also, everytime a file is downloaded, it will be made accessible, so that the next node will be able to download it.

Now that we have stated our actual reflection that we made before making the botnet, let's look at what was the coding part of the project.

III. Part 2

III.1. Something part

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem.

III.2. Something part

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem.

III.3. Something part

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem.

IV. Part 3

IV.1. Something part

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem.

IV.2. Something part

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem.

IV.3. Something part

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem.

Conclusion

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis. Suspendisse interdum ac tellus nec ultricies. Nullam eu bibendum ipsum. Pellentesque in ipsum vel orci ullamcorper malesuada in at turpis. Nulla facilisi. Quisque ullamcorper at sem eget porttitor. Ut sed mi fermentum, fermentum purus in, molestie sem.

Glossary

- **Botnet:** A type of malware which takes control of numerous machines to launch some attacks. Although it's usually used to launch denial of service attacks, it can also be used to crack passwords or even mine cryptocurrency.
- **Other example:** Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque viverra pulvinar dui ut venenatis.

References

- [1] This website is the first example that came to my mind as an example.. <https://example.com>
- [2] This is another example to show peoples what is en entry in the references chapter.
<https://secondexample.com>