Department of Mathematics and Information Technology: Cloud Platform comparison for malware development

Kamil Janowski

Master’s Thesis in Computer Science

Autumn Term 2018

Department of Mathematics and Information Technology

University of Jyväskylä

ABSTRACT

Janowski, Kamil. 2018. Master’s thesis in computer science. University of Jyväskylä. Department of Mathematics and Information Technology (MIT).

The cloud platforms such as AWS, Google Cloud or Azure are designed to cover most popular cases in terms of web development. They provide services that make it easy to create a new user based on his email address, provide tools for inter-service communication, tools to manage the access rights of different users. Malware development however is more of a corner case, where the client application running on the victim’s machine does not have an email address or a google account to authenticate itself and it does not run directly in the cloud, what can make it more difficult to manage the appropriate access rights. Also, the potential attacker may not want to write his own self-contained service, since, especially when managing a large number of clients, it might be much cheaper to run the backend serverlessly.

//TODO: describe research methods (exploratory?)

The following paper explores possible malware backend architectures for different cloud platforms, aiming to optimise the performance, minimize the development time while keeping the code easy to maintain and to minimize the execution cost.

//TODO: write the final conclusion

Keywords: malware, development, cloud, CnC, backend, serverless

CONTENTS

[1 INTRODUCTION 5](#_Toc518731072)

[1.1 Research Problem 6](#_Toc518731073)

[1.2 Research Objective 6](#_Toc518731074)

[1.3 Research Question 7](#_Toc518731075)

[1.4 Key Definition 7](#_Toc518731076)

[1.4.1 Hacker 7](#_Toc518731077)

[1.4.2 Botnet 7](#_Toc518731078)

[1.4.3 Bot 7](#_Toc518731079)

[1.4.4 Serverless computing 7](#_Toc518731080)

[1.4.5 Cloud Computing 8](#_Toc518731081)

[1.4.6 Malware 8](#_Toc518731082)

[1.4.7 CnC server 8](#_Toc518731083)

[1.5 Structure of the thesis 8](#_Toc518731084)

[2 Theoretical Background 8](#_Toc518731085)

[2.1 Common botnet architectures 9](#_Toc518731086)

[2.1.1 Centralised architecture 9](#_Toc518731087)

[2.1.2 Peer to Peer (P2P) Architecture 9](#_Toc518731088)

[2.1.3 Hybrid architecture 10](#_Toc518731089)

[2.2 Common botnet use-cases 11](#_Toc518731090)

[2.3 Command delivery methods 11](#_Toc518731091)

[2.4 File Upload 11](#_Toc518731092)

[3 Methodology 11](#_Toc518731093)

[4 Findings – case study on 3 platforms 11](#_Toc518731094)

[5 Conclusion 12](#_Toc518731095)

[6 Discussion 12](#_Toc518731096)

# INTRODUCTION

The popularity of computing clouds have increased drastically during the recent years. It is perfectly understandable, taken into account that renting the infrastructure from a cloud provider tends to be significantly cheaper than maintaining it inside the company. Things like the rental of the server room, electricity consumed by the servers, cooling of the server room and salaries of people responsible for the maintenance of the servers generate unnecessary overhead in terms of costs of maintenance, which can be drastically reduced when switching to the cloud, while in the same time providing higher availability and better monitoring of the hosted services. Furthermore the cloud providers constantly introduce new solutions allowing to reduce the maintenance costs even further. As we can read in “Serverless Computing: Economic and Architectural Impact” by Gajko Adzic and Robert Chatley (2017, p. 884):

Amazon Web Services unveiled their ‘Lambda’ platform in late 2014. Since then, each of the major cloud computing infrastructure providers has released services supporting a similar style of deployment and operation, where rather than deploying and running monolithic services, or dedicated virtual machines, users are able to deploy individual functions, and pay only for the time that their code is actually executing. These technologies are gathered together under the marketing term ‘serverless’ and the providers suggest that they have the potential to significantly change how client/server applications are designed, developed and operated.

It is important to note however that those technologies are not only available to big corporations trying to lower their cost of server maintenance, but also to hobby software developers and hackers.

A successful attacker may have thousands of devices under his control. In order to control such a large number of devices remotely a highly scalable Command-and-Control (CnC) server is required. Scaling up the virtual machines (VM) however can be costly, while having only a small number of administrators leads to a situation where most of the resources assigned to those VMs are seriously underutilized. While all the remote malware subscribes to the push notification service, it mostly just waits for a command to be generated by an administrator. Effectively, while our CnC server has to be scalable in order to maintain the connection to numerous clients, it requires fairly low computing power until an administrator decides to generate certain load. This suggests that the serverless approach could be applied in this case, what could potentially not only save the attacker a lot of money, but also make such a large scale attack possible in the first place.

## Research Problem

There are many various cloud providers out there. While they all provide services allowing to easily and quickly build secure web applications, the problem of building a CnC server is more of a corner case, that is not necessarily properly addressed by certain clouds. This might yield it impossible to implement such an application in a serverless manner at all, or require to make some compromises and implement workarounds for services that work in a different manner than desired.

## Research Objective

The main objective of the research is to find a way to use the cloud as a CnC server without implementing any application that requires a constantly running server in a Virtual Machine, as those are the main cost generators of the web applications. For this reason we are going to investigate the serverless solutions provided by various cloud providers as well as other services that come with specific clouds that could potentially allow us to set up the communication between the backend and the client application, enable the file transfer, make it easy to manage the access rights of different clients as well as enable the client management in as a whole. We are also going to take a closer look at how the continuous deployment can be solved in various cloud systems.

Each of the approaches will be backed up by a small Prove of Concept (POC) if possible at all. In order to optimise the development time and ensure multi-platform and multi-cloud support of at least parts of our code, all solutions will be implemented with Node.js.

## Research Question

When focusing on various cloud platforms, such as Amazon Web Services (AWS), Google Cloud Platform (GCP) and Azure the approach to the problem might be completely different and the cost of execution may different significantly as well. The question in this case is, which one of the platforms is the best suited and the cheapest to run our CnC server.

## Key Definition

### Hacker

Hacker is a malicious attacker attempting to access resources of a remote machine. In this thesis the term “hacker” will be used to describe the administrator of the CnC server and in the same time the administrator of the botnet.

### Botnet

A botnet is a network of private computers infected with malicious software and controlled as a group without the owners' knowledge, e.g. to send spam.

### Bot

A bot in this case is a single client application executing (and in some architectures issuing) the commands on the infected device.

### Serverless computing

The “serverless” computing is a marketing term that relates to developing single functions, rather than a large monolithic application and then being charged only for the actual execution time of the function, rather than for the constantly running server that technically is still there, but is hidden from the service user. The concept was originally introduced by Amazon in their AWS cloud in 2014 under the name of Lambda. Since then all major cloud providers introduced various equivalents in their solutions. As many instances of lambda can be triggered in parallel, this solutions is not only cheaper to execute, but also potentially infinitely scalable. This is why it’s commonly used for a wide range of applications, starting with REST API call processing and ending with Big Data event handling.

### Cloud Computing

As Amazon defines it on <https://aws.amazon.com/what-is-cloud-computing> (24-06-2018):

Cloud computing is the on-demand delivery of compute power, database storage, applications, and other IT resources through a cloud services platform via the internet with pay-as-you-go pricing.

### Malware

Malware, or malicious software, is any program or file that is harmful to a computer user. Malware includes computer viruses, worms, Trojan horses and spyware. These malicious programs can perform a variety of functions, including stealing, encrypting or deleting sensitive data, altering or hijacking core computing functions and monitoring users' computer activity without their permission.

### CnC server

In “Survey on botnet: its architecture, detection, prevention and migration” by Ihsan Ullah et al. (2013) the CnC servers are defined as centralised servers allowing the malicious attacker to remotely control a number of clients applications that connect to it.

## Structure of the thesis

//TODO: finish

# Theoretical Background

## Common botnet architectures

As we can read from “Survey on botnet: its architecture, detection, prevention and migration” by Ihsan Ullah et al. (2013, p. 661-662), as well as “Botnet Communication Patterns” by Gernot Vormayr et al. (2017, p. 2772) there’s a number of different architectures that can be developed depending on the attacker’s needs.

### Centralised architecture

The architecture assumes that there’s one CnC server that all the clients can connect to. It tends to use either Internet Relay Chat (IRC) or HTTP as the communication protocol. This solution tends to be the most commonly seen due to the ease of implementation as well as high efficiency. The main drawback of the approach is that it is fairly easy to detect. Each of the clients of the botnet needs to have a hard-coded address of the server that it is going to communicate with. Effectively simply editing the byte code of the application (or decompiling it, if possible) allow you to quickly read the address of the CnC server and then block all the traffic to it. The address can also be seen through network sniffing. This problem however can be mitigated through the use of Domain Generator Algorithms (DGA).

DGAs generate different domain names based on a changing input. For instance, a different domain could be used based on the current time. This then requires all clients to have synchronized time down to one hour. While relying on the system time might not necessarily be a good idea, as the system time largely depends on the user-specified settings, it can be easily achieved by polling popular websites that contain such information.

### Peer to Peer (P2P) Architecture

The approach allows to hide most of the network traffic by introducing the supervisor-bot, who becomes responsible for delivering the command to other clients, who later on can forward the command even further. While the source of the command becomes fairly difficult to detect in this case, the actual delivery as well as the delivery of the result takes significantly more time than in the centralised architecture. This makes such botnet difficult for the attacker to manage. Also, it is important to note that the architecture is prone to the Sybil attack, where the attacker subverts the reputation system of a P2P network by creating a large number of pseudonymous identities, using them to gain a disproportionately large influence.

In a fully meshed botnet every client is linked to every client. This way it is possible to reduce the latency as well as ensure that the removal of any number of bots does not disrupt the communication. This solution however is not scalable due to the number of required connections in larger botnet. Additionally, the larger number of connections increases the visibility of the botnet. Also adding or removing a single client generates a significant network traffic as all other clients have to register the information about the new bot.

The topology unfortunately is difficult to implement due to the challenges of finding the initial peers and reliably distributing commands to every bot.

The list of peers can be hard-coded directly in the executable or provided by a cache server. The first solution however can work only in a very targeted attack and should the botnet be detected, the list can be easily extracted from the code.

In the second case, the server is visible to the public internet and that brings back all the issues related to the centralised architecture.

Finally, depending on the NAT configuration, not every computer has direct access to the internet making it difficult to access from external network.

### Hybrid architecture

Hybrid architecture combines both centralised architecture and the P2P one. Instead of bots connecting directly to the CnC server, an additional proxy layer consisting of bots connected in a P2P topology is added. Determining whether a certain bot should behave only as proxy or P2P accessed worker can be done based on the connectivity properties (such as when some of the infected devices don’t have the direct access to the CnC server). In order to lower the probability of detection of the CnC server, additional layers of P2P connection can be added, although that comes with the cost of increased latency.

## Common botnet use-cases

//TODO: F-secure has some report on it. Same goes probably for every antivirus company…

## Command delivery methods

How do you do it in P2P and how in CnC. Explain why CnC approach is faster and more convenient. IRC protocol? MQTT protocol?

## File Upload

//TODO: in clouds there will be buckets. Otherwise you upload stuff through IRC or HTTP directly.

# Methodology

What you’re going to use to research your paper.

Describe that you’re going to use the qualitative research method and shit.

Describe the study framework. (check the link to that research from Nguyen, 7.2.3 – Study Framework)

# Findings – case study on 3 platforms

Market overview of cloud computing / 3 platforms. All the info about all 3 platforms. (in separate sub-points)

Comparison of the platforms.

# Conclusion

# Discussion

references

appendices