Detection of Botnet Fast-Flux Domains by the aid of spatial analysis methods

by

Felix Stornig

2nd Bachelor-Thesis

for the conferring of the academic degree of
Bachelor of Science

Carinthia University of Applied Sciences
Studiengang Geoinformation und Umwelttechnologien

Supervision:

Dipl.-Ing. Dr.techn. Victor Manuel García-Barrios
Carinthia University of Applied Sciences

DI Martin Deutschmann
TECHNIKON Forschungs- und Planungsgesellschaft mbH

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Bachelorarbeit selbstständig verfasst und alle benutzten Quellen und Hilfsmittel angegeben habe. Wörtliche
und sinngemäße Zitate sind als solche gekennzeichnet.

Villach, [15.06.2013]

Felix Stornig

Acknowledgement

I want to thank both my supervisors, at the Carinthia University of Applied Sciences and at the TECHNIKON Forschungs- und Planungsgesellschaft mbH, as well as the staff of the latter, for their support and assistance in writing this thesis and during my internship in general. I want to thank my colleagues, friends, parents and every other person who supported me during this time.

Abstract

Recently, botnets emerged as one of the biggest threads to businesses in terms of cyber-security. Basically botnets are a network of hosts that have been infected by malware which enables the controller of botnet utilize these hosts as a distributed platform for commencing various criminal operations, such as launching various forms of attacks on networks, including Distributed Denial of Service (DDoS) attacks, and password fishing. Remote control of the infected hosts that are part of a botnet is maintained by so-called 'command and control' (C&C) channels. These days, communication over these C&C channels is maintained over numerous domain names that are associated with hosts on the botnet. Each of these botnet-domains can in turn be associated with numerous IP-addresses which can be rapidly changed, in order to complicate the tracing and disabling of a botnet. The mechanism of rapidly changing the IP-addresses that are associated with a particular domains name, is known as 'Fast-Flux' and is not only used in the context of botnets, but also by large content distribution networks (CDNs) like 'google.com, for example.

The subject of this work is to investigate a relatively new attempt in detecting Fast-Flux domains that are involved in unlawful activities, which utilizes methods of geoinformation and spatial statistics, as well as to discuss the results that can be achieved by employing them, in order to give a statement about their validity.

Zusammenfassung

Botnets haben sich inzwischen als eine der größten Bedrohungen für Gewerbe aller Art hervorgetan was Cyber-Security anbelangt. Im Grunde genommen ist ein Botnet ein Netzwerk von mit Malware infizierten Rechnern, die von demjenigen der das Botnet kontrolliert dazu benutzt werden können verschiedenste kriminelle Operationen durchzuführen. Dies beinhaltet verschiedene Formen des Angriffs auf Netzwerke, wie zum Beispiel Distributed Denial of Service (DDos) Attacken oder Phishing (Password Fishing). Die infizierten Rechner werden über sogenannte ‚Command and Control' (C&C) Kanäle ferngesteuert. Heutzutage findet die Kommunikation über diese C&C Kanäle über eine Vielzahl von Domain-Namen statt, die mit Rechnern in einem Botnet assoziiert sind. Jede dieser Botnet-Domains kann wiederum mit einer Vielzahl von IP-Adressen assoziiert werden welche sich in schneller Abfolge ändern können, um das Verfolgen und lahmlegen von Botnets zu erschweren. Die Vorgehensweise, IP-Adressen die mit einer Domain assoziiert sind in rascher Abfolge zu ändern, wird ‚Fast-Flux' genannt und wird nicht nur von Botnets, sondern auch von großen Content Distribution Networks (CDNs), wie zum Beispiel ‚google.com' genutzt. Der Kern dieser Arbeit dreht sich darum einen relativ neuen Ansatz zum Aufspüren von in illegale Aktivitäten verwickelten Fast-Flux Domains, der Gebrauch von Mitteln der Geoinformation und Räumlichen Statistik macht, zu untersuchen und die damit erzielbaren Ergebnisse zu erörtern und dann ein Urteil über ihre Aussagekraft zu fällen.

Contents

1 Introduction

In recent times, special forms of malware, so-called bot-nets, have emerged as one of the most serious Internet threads to all kind of businesses, companies as well as end-users and their networks respectively.

According to Feily et al. (2009) and Nair and Ewards (2012) a bot-net is a network formed by malicious pieces of software that are called "bots". These "bots" infect computers and hide themselves on the system while operating unnoticed by the user. As stated in Zhang et al. (2011), these bot-nets are then subsequently used to launch network attacks, DDoS[1] attacks as well as information phishing[2] attacks.

In February 2013 an Europe-wide cyber security project with the acronym ACDC[3], which stands for Advanced Cyber Defence Infrastructure, has been started. This project, made up of the contributions of 28 partner institutions and companies from 14 different countries, aims to fight botnets by addressing issues of identification, analysis, prevention, mitigation, recovery and evaluation of those. The work done during the project shall result in a number of anti-botnet services of different kinds, which are to be operated by the member states. One of these services comprises of the so-called 'centralised data clearing house', which will act as a central information hub for anti-botnet actions, where data from a wide range of different sources will aggregated in a common data format, stored and analysed. The research done during this thesis happens in the context of the ACDC project, as a theoretical contribution to the complex matter of how botnets could be detected and mitigated and will incorporate methods of geoinformation and spatial statistics respectively.

1 Distributed Denial of Service
`http://oxforddictionaries.com/definition/english/DDoS` [last access: 14th of June 2013]
2 `http://oxforddictionaries.com/definition/english/phishing` [last access: 14th of June 2013]
3 `http://ec.europa.eu/information_society/apps/projects/factsheet/index.cfm?project_ref=325188` [last access: 14th of June 2013]

1.1 Motivation and Problem Definition

Information as well as commands that control the network are passed to and from the infected computers to the controller of the bot-net (the so-called bot-master) by means of a command and control (C&C) channel by which the bot-master can update and direct the botnet. The architecture of these channels may span over a range of network topologies and utilize different protocols for communication. (Zhang et al. 2011)
As Nair and Ewards (2012) stated these channels originally relied on IRC[4] or HTTP[5] technologies which could be disabled with relative ease. Nowadays, amongst others, peer to peer (P2P) technology is used to establish communication between bots, with C&C functionality distributed between many command and control servers that are interconnected. P2P technology has the advantage over IRC or HTTP based channels that it reasonably complicates to shut down or monitor them. (Feily et al. 2009)
As attempts were made correspondingly in order to disable or shut down particular nodes on the bot-net respectively, bot-nets began to utilize a network mechanism that is called Fast-Flux. At a glance, Fast-Flux enables bot-nets to associate numerous IP-addresses with domain names that are used for communication among them. Those associations can be switched in a rapid manner, a process that was originally used by legitimate websites for the sake of maintaining availability and load-balancing (see subsection 2.1.2). (Stalmans et al. 2012)

The malicious nature of bot-nets demands a reliable mechanism for detecting and disabling them. A major step in this context is to reliably detect domains that are using Fast-Flux for the sake of obscuring botnet communication, in order to disable them. Unfortunately, genuine detection methods are commonly complex and demand a lot of resources, such as comprehensive traffic monitoring. (Nair and Ewards 2012)
Another problem in correctly identifying Fast-Flux domains however, is that legit domains could also expose Fast-Flux behaviour as they employ it for load-balancing reasons. A proposed detection mechanism therefore should be also capable of distinguishing between legitimate and malicious domains employing Fast-Flux mechanisms. (Stalmans et al. 2012)

4 Internet Relay Chat
`http://oxforddictionaries.com/definition/english/IRC` [last access: 14th of June 2013]
5 Hypertext Transfer Protocol
`http://oxforddictionaries.com/definition/english/HTTP` [last access: 14th of June 2013]

For the sake of clarification, the terminology that will be used further throughout this work (which has been determined on the basis of the terms used by Stalmans et al. (2012), Feily et al. (2009) and Wang et al. (2012)) is organized as follows:

- Domain names that have a Fast-Flux mechanism standing behind them will be henceforth designated as **Fast-Flux domains (FFDs)**.
- Domains that are recognized to be not involved in Fast-Flux activities regardless of if they are malicious or not are simply called **Non-Fast-Flux domains**.
- Domains that are recognized and trusted, which for example include google.com or twitter.com, will be designated as **legit**.
- Domains that are recognized to be involved in bot-net activities or to distribute malware respectively, will be designated as **malicious**.
- Domains that are not explicitly known to belong to any of the previous kinds, will be referred to as **unclassified** or **mutually malicious**.
- Domains that are indicated by a statistical test to be using Fast-Flux will be termed **suspected** and otherwise **non-suspected**.

Additionally it is important to clarify that whether a domain uses Fast-Flux or not, can not be inferred from whether a domain is legit or malicious alone.

This work deals with a relatively new attempt in identifying Fast-Flux domains as well as distinguishing legit Fast-Flux domains from malicious ones, by employing methods of spatial-statistics and statistic analyses. The basis for this approach is supplied by the research done by Stalmans et al. (2012) and Wang et al. (2012).

Stalmans et al. (2012) has shown that this approach which is using spatial analysis methods has the advantage that the statistical classifiers are lightweight and not as resource demanding as traditional ones in terms of generated network traffic. Also the delay between a domain registered and the said domain to be classified is close to zero.

The main objective of this work is to combine classification methods from the aforementioned authors and investigate the results that can be achieved by combining them, as well as to issue a statement about the validity of those methods.

The next chapter will give an overview of how this is going to be achieved.

1.2 Method of Solution

The detection approach of of Fast-Flux domains (FFDs) utilizing spatial statistics centres around the idea that IP addresses of FFDs can be mapped at an approximate (but coarse) physical location on the planet. These locations, when analysed by methods of spatial statistics, exhibit a particular spatial distribution and behaviour respectively that is significantly different from those of domains that do not employ Fast-Flux services as well as that this behaviour varies over time differently for legitimate and malicious domains.(Stalmans et al. 2012)

The procedures necessary for the statistical analyses will be implemented in the R scripting-language, which is a free and widely-used environment for statistical computing, as well as for developing statistical software. (R Foundation 2013) The reasons for utilizing R during the course of this work include: (Antonio Gasparrini 2011)

- (as mentioned) R is free software
- it features reasonable performance
- R features a large community and is well documented

The data necessary for the statistical tests to be conducted basically consists of sets of domain names which behaviour is to be investigated; this datasets will consist of both, names of trusted and widely known domains, as well as domains that are suspected/reported to be part of a botnet or distribute malicious software. All of this data will be obtained from publicly available sources. Domain name information will include both, widely trusted (seomoz.org 2013) as well as suspicious malware domain names (malwaredomains.com 2013) and domain names that are suspected to be associated with a real botnet (abuse.ch 2013).

Domain related IP-information will be entirely obtained by using the network administration command-line tool 'dig' which counts to the standard software repertoire of some linux-distributions, and is used to retrieve information from domain name system (DNS) servers. (linux.die.net 2013)

Spatial information of IP-addresses, namely IP geolocations will be obtained by using the database supplied by the free and open-source 'freegeoip.net' Web service, whose source code can be used to set up a local instance of the database. (Alexandre Fiori 2013)

The discussion of the results will incorporate a detailed examination of the statistical outcomes that have been achieved, by considering different statistical values in their respective context.

1.3 Expected Results

The following outcomes are expected from this work:

- a statistic workflow that processes a list of domain names and classifies these domains as either suspected or non-suspected, as well as a server-runnable R-script that implements the said workflow (see chapter 3)
- a subsequent examination of the results and a discussion about the validity of the classification methods (see chapter 4)

1.4 Structure

The following chapters of this work are organized as follows:

- The theoretical background will be discussed in chapter 2 of this paper; it will serve to clarify and argue the principles that stand behind this in a more detailed manner. The topics presented in this chapter will include: Fast-Flux domains and their issues in the context of spatial statistics, their spatial behaviour as well as other technical topics such as the R scripting language, amongst others.
- Chapter 3 of this paper will present the methodology of the work that was performed in the context of this thesis. It will detail the main workflow and its implementation during the course of this work as well as related tasks such as data gathering and how the statistical classifiers were realised.
- The outcomes of this work will be interpreted and discussed in chapter 4, by going into their statistical values and key observations; also the performance of the statistical classifiers will be investigated.

- Chapter 5 will give a conclusion about what has been achieved, discuss the validity of the methods applied and summarize the results. Additionally it will give a summary about the lessons-learned as well as some final remarks and related future prospects.

2 Theoretical Background

This chapter explains the theoretical background topics that are necessary to be understood in order to deal with the details of this work, as well as with topics originating from research that was carried out by aforegoing authors, from which the research done in this context draws its theoretical basis.

2.1 Fast-Flux Domains and the Domain Name System

As the introductory chapter gave a preliminary description of what Fast-Flux domains (FFDs) are, this section will explain them in a more detailed manner. Prior to that, a short explanation of the Domain Name System (DNS) will be given as it is closely related to how Fast-Flux is functioning.

2.1.1 The Domain Name System (DNS)

The Domain Name System (DNS) is a decentralised, hierarchical database system that enables the mapping of a domain e.g. 'mail.fh-kaernten.at' to an IP-address that is associated with this domain e.g. '193.171.127.154'. DNS enables the user to employ a recognizable name when accessing a web-site on the internet rather than using its IP-address which might be complicated to remember. The theoretical process of resolving domain names into IP-addresses works roughly as follows:
(IBM 2008)

1. A DNS query starts by contacting a root name sever which is usually preconfigured on the querying host. The root name server then returns the IP-address of the name-server responsible for the top-level domain of the domain queried e.g. '.at'

2. Then top-level domain server is contacted and finds the IP-address for the second-level domain e.g. 'fh-kaernten'
3. The previous step is repeated until all other name spaces of the queried domain consists of (an example would be 'mail' in 'mail.fh-kaernten.at') have been processed and the IP-address of the wanted host is returned.

Figure 2.1 illustrates this workflow on the example of 'mail.fh-kaernten.at'.

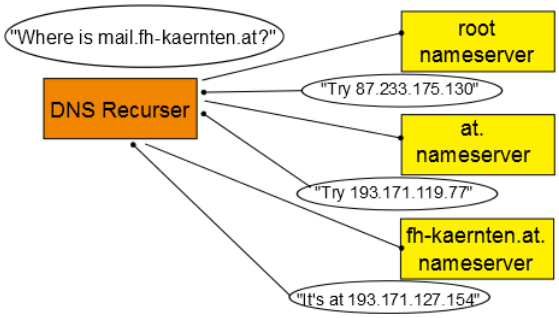

Figure 2.1: Theoretical DNS domain name resolution

Beyond that it has to explained that a DNS database record that is held by a DNS server consists of several different sections, which store different types of information. For the course of this work, however only two of them will be relevant for us (IBM 2008):

- the 'A' record ('Address Mapping' or 'Answer' Record) which contains the IP address of a specific domain name to be resolved
- the 'NS' record ('Name Server' Record) which specifies IP-addresses for other name servers who respond to a specific domain name queried

The following snippets show sample outputs from the 'dig' command line tool obtaining DNS record information for the domain 'fh-kaernten.at':

The following snippet displays the Answer- or A-record for the domain; the corresponding IP-address is found below the heading 'ANSWER SECTION', right of the domain name.

```
; <<>> DiG 9.3.2 <<>> fh-kaernten.at a
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1076
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;fh-kaernten.at.                        IN      A

;; ANSWER SECTION:
fh-kaernten.at.         2229    IN      A       193.171.127.159

;; Query time: 77 msec
;; SERVER: 192.168.0.1\#53(192.168.0.1)
;; WHEN: Thu May 02 12:24:59 2013
;; MSG SIZE  rcvd: 48
```

This snippet on the other hand shows the respective NS-record below the heading 'ADDITIONAL SECTION', the IP-addresses belonging to the domains's responding name servers are again located right aside the name servers' domain names. It can also be observed that these two name servers are associated with the domain 'fh-kaernten.at' in the 'ANSWER SECTION' of the request.

```
; <<>> DiG 9.3.2 <<>> fh-kaernten.at ns
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1817
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 2

;; QUESTION SECTION:
;fh-kaernten.at.                        IN      NS

;; ANSWER SECTION:
fh-kaernten.at.         3600    IN      NS      ns1.fh-kaernten.ac.at.
fh-kaernten.at.         3600    IN      NS      ns2.fh-kaernten.ac.at.
```

```
;; ADDITIONAL SECTION:
ns1.fh-kaernten.ac.at.  3600    IN      A       193.171.119.77
ns2.fh-kaernten.ac.at.  3600    IN      A       193.171.127.177

;; Query time: 77 msec
;; SERVER: 192.168.0.1#53(192.168.0.1)
;; WHEN: Thu May 02 13:03:00 2013
;; MSG SIZE  rcvd: 115
```

For the sake of completeness these two outputs are displayed in all detail, although only the above mentioned pieces of information are relevant here.

The data obtained from a domain's A and NS record(s) will matter when considering parameters for the spatial behaviour of legit and malicious domains which will be explained in section 2.3. Further technical details of how DNS works exactly will not be relevant in the current context[1].

2.1.2 Fast-Flux Domains (FFDs)

The exact technical details of how Fast-Flux networks operate will be only of limited importance here, as the spatial distribution and behaviour of fluxed IP-addresses (which will be explained in 2.3) will be essential for the course of this work rather than the technology itself.

As mentioned in subsection 2.1.2, the main function of Fast-Flux domains is to associate many, often as many as thousands of IP-addresses with one single domain name and flux (change) them rapidly which also finds a legitimate application in respect to load-balancing matters, in order to ensure availability on web sites that face a high volume of requests each day. Hereby changing IP-addresses with high frequency is achieved by changing their corresponding DNS-records.

There are three basic types of Fast-Flux operation (Caglayan et al. 2009):

1 for details see `http://pic.dhe.ibm.com/infocenter/iseries/v6r1m0/topic/rzakk/rzakk.pdf` [last access: 4th of June 2013]

- Basic Fast-Flux or Single-Flux: here, only the IP-addresses of a specific domain which employs Fast-Flux are fluxed over time
- NS fluxing: in this variant the IP-addresses of name servers that are responding to a specific FFD
- Double Flux: this type of Fast-Flux operates by fluxing both the IP-addresses of a FFD itself as well as the IP-addresses of their responding name servers which provides an additional level of redundancy

According to Zhao and Traore (2012), Fast-Flux networks, regardless of type, expose a common set of characteristics which are certain for them (selection):

- Low Time-to-live:
  The time-to-live (TTL) value indicates how long a particular DNS record for a particular domain should be kept by the name server before it is discarded and updated, hence a short TTL is essential for FFDs in terms of getting sure that DNS records for an FFD always point to the most up-to-date network nodes. While TTL is normally quite long for Non-Fast-Flux web-sites (about 24 hours) it is proven to be rather short for FFDs (close to 5 minutes).
- Number of unique A records:
  A significant difference between IP-addresses returned from legit and malicious domains can be observed when examining the number of IP-addresses returned by subsequent DNS queries over some time. While legit domains generally employ a small, designated set of addresses for their services, malicious Fast-Flux networks may utilize ten thousands of IP-addresses which change over time.
- IP Networks:
  The IP-addresses that are employed by legitimate domains are usually located within a common address range, and thus normally show some correlation. The addresses of malicious domains on the other hand may be arbitrarily distributed over the range of all possible IP-addresses, a trait that is also observable in a similar way when examining the IP geolocations of such address sets.
- IP Geolocation:
  Analogous to the point above IP-addresses of malicious FFDs will be geographically dispersed more arbitrarily, while those of legit domains are generally clustered

in certain areas and distributed following some geographical intelligence; the IP geolocation will serve as the basis for the statistic classifiers that are investigated during in the scope of this work (see section 2.3).

As stated by Zhao and Traore (2012) Fast-Flux domains can be easily mistaken for legit domains employing load-balancing mechanisms, when it comes to analysing their behaviour. An analogous technique that is used for such a purpose is Round Robin DNS, where IP-addresses of a domain are simply rotated over time to maintain availability and prevent traffic bottlenecks. This is especially utilized by so-called Content-Distribution-Networks (CDNs), which distribute their service over numerous serves to provide them to the end-user with high performance and high availability. A known example of a CDN would be 'google.com'; the time-to-live TTL of the DNS-records of google.com can be as short as for those of a Fast-Flux domain.

2.2 IP geolocation at a glance

IP geolocation is the mapping of an IP address to a physical, geographic location. Amongst other things, for example, advertisers and search engines employ this service in order to serve country specific contents to end users. There are two main approaches in this (Gill et al. 2010):

- measurement-based geolocation
- database-based geolocation

The former approach relies on different algorithms which examine monitored network traffic in order to make inferences about an IP address's location, which can be quite complex; details about how this works will not matter within the scope of this work. The latter and, in the context of this work, more interesting approach uses databases of location mappings which can be publicly available as well as proprietary ones. Public databases are mostly provided by regional Internet registries, such as ARIN[2] or RIPE[3]. Proprietary databases include the ones supplied by companies such as Quova or Maxmind for example.

2 American Registry for Internet Numbers
`http://www.arin.net` [last access: 14th of June 2013]

3 Réseaux IP Européens
`www.ripe.net` [last access: 14th of June 2013]

The IP geolocation data that will be employed during the course of this work is delivered indirectly (as geolocation data will be fetched by a local FreeGeoIP service instance - see section 1.2) by Maxmind, which also offers a free geolocation database service[4]. Regardless of approach, the accuracy of contemporary geolocation algorithms is limited to a range of about 35-194 kilometers. (Gill et al. 2010)

2.3 Spatial behaviour of Fast-Flux and Non-Fast-Flux domain IP-addresses

As already stated previously in this paper, IP-addresses of a domain can be mapped to a coarse real world location by using an IP-geolocation service. Those geolocations exhibit a spatial distribution behaviour that is different for Fast-Flux and Non-Fast-Flux domains which arises from the characteristics that were described in the previous section 2.1.2. The geographically more and less systematically dispersed nature of IP-geolocations of FFDs can serve to define a set of spatial classifiers that, considered together with other, non-spatial attributes could be used to classify a domain name as either legit or malicious with a sophisticated classification accuracy. In the course of this work, two spatial classifiers in particular will be proposed and incorporated.

The first classifier will be the index of spatial autocorrelation, more precisely: the Moran Index of Spatial Autocorrelation which has been suggested as a high quality classifier in the research done by Stalmans et al. (2012). Spatial autocorrelation expresses how points within a two-dimensional space or a geographic area respectively are dependant on each other and is based upon the principle that the attributes of points that are closer together expose a greater degree of similarity than points that are farther away from each other. Spatial autocorrelation can be negative, positive or zero. Positive autocorrelation can be observed when points with similar attribute values are near each other, in contrast a negative autocorrelation can be seen when similar attribute values lie farther away from each other on the one hand and more distinct values lie closer together on the other hand. A spatial autocorrelation of zero indicates that the values of point attributes are distributed in a spatially random manner. The Moran Index returns values in the range from -1 to +1 from a given set of points with a particular attribute specified to be

4 `http://dev.maxmind.com/geoip/legacy/geolite` [last access: 14th of June 2013]

examined, whereas negative values indicate a negative spatial autocorrelation and visa versa; a Moran Index of 0 also indicates a spatial autocorrelation of zero, and thus spatial randomness.

The Moran Index can be calculated by using the following formula (Griffith 2009):

$$I = \frac{n}{S_0} \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(x_i - \bar{x})(x_j - \bar{x})}{\sum_{i=1}^{n} (x_i - \bar{x})^2},$$

and

$$S_0 = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}.$$

where:

- I represents the Moran Index
- n is the number of observed points
- x_i represents the nth variable's value (the examined attribute of a point)
- $\bar{x}$ is the mean of x
- w_{ij} is the weight (distance) between the two points i and j
- S_0 is the sum of all w_{ij}s

As it can be observed from an example given by Stalmans et al. (2012) which compares the IP-geolocations of google.com and a malicious bot-net domain on a world map (depicted in figure 2.2), those geolocations that are related to the bot-net domain (depicted as circles) are geographically distributed over the whole world, whereas google.com IP-addresses (depicted by a triangle) are all concentrated on one point (Mountain-View, California, USA). Stalmans et al. (2012) tested the results of Moran Index calculations by using several different spatial attribute values, such as time zones and UTM[5] / MGRS[6] coordinates, with

5 Universal Transverse Mercator `http://geology.isu.edu/geostac/Field_Exercise/topomaps/utm.htm` [last access: 14th of June 2013]

6 Military Grid Reference System `http://earth-info.nga.mil/GandG/coordsys/grids/universal_grid_system.html` [last access: 14th of June 2013]

MGRS coordinates achieving the highest rate of true positives.

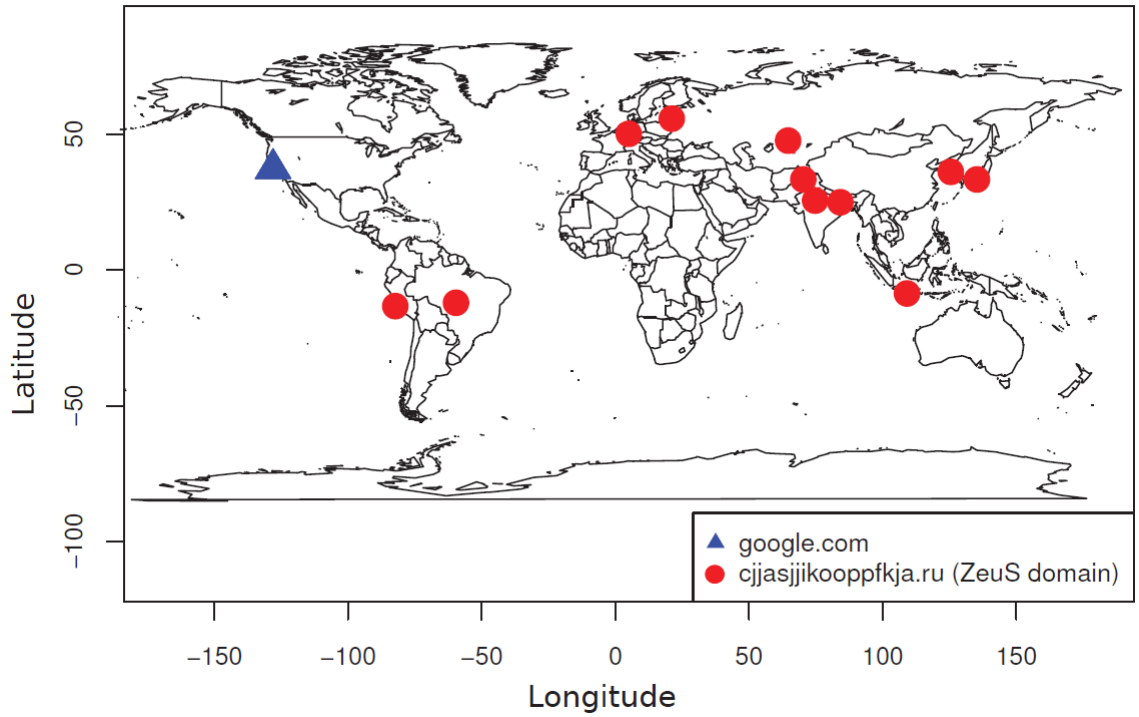

Figure 2.2: Differences in the Geographic distributions of legit and malicious domain IP-geolocations (Stalmans et al. 2012)

This works approaches the same principle using latitude / longitude coordinates of IP-geolocations associated with a particular domain. When the coordinates of a point are used as attribute values within Moran Index calculations, Stalmans et al. (2012) asserted that the results exhibit a characteristic distribution with indices of Non-Fast-Flux domains scattering around 0 and those of Fast-Flux domains scattering around 1. This observation could be repeated during the tests performed during this research, as can be seen in chapter 4.

The second important spatial classifier that will be utilized during this work is the so-called spatial service distance, as it has been proposed by Wang et al. (2012). Basically the spatial service distance is defined by the average distance between the geolocations of the IP-addresses associated with a domain name and the geolocations of the IP-addresses of the name servers that are responding to this domain name. For Non-Fast-Flux domains average service distance is often (but not always) close to zero with their geolo-

cations often all being concentrated in or around the same area. For FFDs however, this service distance may become quite large, which is rooted in the fact that an FFD's IP-geolocations (and thus those of their name servers) are commonly dispersed arbitrarily over a great area. Therefore a service distance value is able to act as a spatial domain classifier.

2.4 A brief outline of the R scripting language

The purpose of this section is to give a brief outline of a few wisely selected topics regarding the R environment which are relevant for understanding the more detailed descriptions of the script which will be given in chapter 3.
As stated in section 1.2 R is a freely available scripting language for statistical computing. R can be run from command line (by the 'R' command) as well as by using a window system. When working on the command line, R can be used either interactively or by simply calling scripts by directing input files to the 'R' command. Using R interactively means that a prompt is given to the user which then types in or calls R commands directly. During the course of this work however, all scripts will be run non-interactively by supplying input files. The 'R' command can take several input parameters only one that needs to explicitly mentioned here is the '–args' parameters which can be used to supply command line parameters of any form to be further used inside the run script (a sample call can be see in subsection 3.2.2).
Whenever R is run (without regard to if this is done interactively or not), this is referred to as having an 'R session'. When using R objects are created as well as stored during a such session; the collection of objects created is called the 'workspace'. This workspace can be saved to a compressed workspace file during a session and can be restored in another to continue working on it. (Venebles and Smith 2013) Further, it has to be said that R functions and their belonging data are organized in the form of packages. These packages act as libraries for the R environment and are maintained by a central repository[7]. R packages can be installed and loaded entirely by using the command line or calling the necessary commands within a script respectively. (Venebles and Smith 2013)
At last it is necessary to give a short outline of the most basic data structure of the R

7 `http://CRAN.R-project.org/` [last access: 14th of June 2013]

langauge, the vector. In 'R' the vector is a indexed set of values which can be (amongst others) of the following types:

- integer - an integer real number
- character - a single character (a vector of characters is basically a string)
- logical - a boolean value either designated as 'TRUE' or 'FALSE'

This will matter when describing the implemented script's parameters in subsection 3.2.2. (Venebles and Smith 2013)

The next chapter deals with the methods, processes and implementations that have been performed during the course of this work.

3 Methodology

This section describes how the objectives of this work have been realised in detail, such as the tasks that had to be performed during the implementation of the actual worflow and prior, as well as subsequent ones.

3.1 Preliminary tasks and data gathering

The main tasks that had to be carried out prior to the actual implementation include:

- Research on theoretical topics
- Selection of a processing environment
- Selection of methods for gathering information that will be needed during processing
- Gathering and selection of data to be tested

Preliminary research was made up of both, information from commercial and none-commercial sources. Research regarding specific fields, such as botnet-detection including the works of Stalmans et al. (2012) and Wang et al. (2012) have been obtained from the IEEE[1] Digital Library; other topics such as general statistics and spatial statistics and material relating to the R scripting language have been obtained from documents that are freely available to the public.

R was selected as processing environment for the reasons that have already been described in section 1.2. Although the first versions of the implemented script ran on a Microsoft

1 Institute of Electrical and Electronics Engineers
`http://ieeexplore.ieee.org/Xplore/guesthome.jsp` [last access: 14th of June 2013]

Windows 7[2] desktop machine, the script was sourced out later on to a dedicated system in form of a linux[3] server, for security reasons as it will be detailed in subsection 3.2.1.

The data that was gathered in the preface of the actual implementation work mostly included publicly available lists of domain names which are deemed to be used for once feeding the script. As stated previously in section 1.2, this data consisted of legit domains as well as domains that were reportedly involved in botnet activities or distributing malware. Here it is important to say, that although the latter type of domains are reported to be involved in a botnet, these domains do not necessarily employ fast-flux mechanisms (see chapter 4). The reason for selecting data from both groups was to supply a control group of domains to a list of reported botnet domains for the statistical test performed during execution of the script. The search for data was basically made up of a net search after reporting sites for both types of domains. After examining several potential data sources and the data available at them, the following quality criteria have been assumed in order to narrow down the data available to a set that will finally used in a productive manner:

- up-to-dateness: the data shall be no older than 6 months and/or be updates on a regular basis
- free availability: the data shall be available without the necessity of payment
- format: the data shall be available either in CSV[4] format or shall be easily reformatable into CSV as CSV has been selected as the data format to be used when supplying data to the script

The list of reportedly legit domains is basically a list of the top 500 registered domains ranked by the number of domains which link to them. This list was obtained from the social media monitoring site 'seomoz.org', which updates this list every month. (seomoz.org 2013)
A list of reported botnet domains has been obtained from the Swiss security blog 'abuse.ch' which features a tracking site for the so-called 'ZeuS' botnet and offers a list of associated

2 `http://en.wikipedia.org/wiki/Windows_7` [last access: 14th of June 2013]
3 `http://en.wikipedia.org/wiki/Linux` [last access: 14th of June 2013]
4 Comma Seperated Values
`http://dictionary.reference.com/browse/comma%20separated%20values` [last access: 14th of June 2013]

domains for the sake of blocking them in firewalls; this domain block-list currently contains of 854 domain names and is update continuously (abuse.ch 2013).

In addition to data that was obtained from the said sources, the script was equipped with the capability to generate a given number of 'simulated' botnet domains whose IP addresses are selected randomly with an overall arbitrary spatial distribution.

3.2 Main workflow

The following subsections describe the details of the main workflow, its implementation in R and how the statistic classifiers were realized, as well as further issues that arose with it.

3.2.1 Processing environment and security risks

During the course of the work two different operating systems have been deployed, for R to be run on:

- OpenSUSE Kernel 12.1
- Ubuntu Server 12.10 "Quantal Quetzal"

The reason for selecting the former of the two was the availability of a ready to use system right at workplace. Here a test setup of the script and all other necessary components have been implemented and the first function tests have been conducted. As the work reached a state where the potential interference with presumed malicious domains became unavoidable, the whole setup have been outsourced to an external virtual server with root access, for not taking any security risks.

3.2.2 Structure and implementation

Before the actual workflow is explained, it has to be said that the implementation as a whole consists of a number of components that are located within a common folder and interact together during runtime. This root folder is designated as 'DBFFD-SAM' wich is

basically an acronym for this work and stands for **Detection of Botnet Fast-Flux Domains (by) Spatial Analysis Methods**. The components that are contained within can be outlined as follows (a graphical outline is given in figure 3.1):

1. Data Folder:
   This folder contains the input data files for the script in CSV format.
2. SQLite GeoIP database:
   This is the local instance of the GeoIP SQLite database from which the IP geolocations are selected via the DB Search script.
3. Python DB Search Script:
   This is a pyhton script that searches the database for the geolocation that is belonging to an IP address which has to be supplied as parameter for the script; it is shipped with together the GeoIP database.
4. Python DB Update Script:
   This script downloads updates for the database from maxmind.com, it is also responsible for initializing it prior to first run; this script also comes together with the database.
5. DBFFD-SAM Main Script:
   This is the script that is responsible for executing the main workflow and for conducting all data processing, statistical tests and other related tasks. It is the main entry point for execution that is started first.
6. R Function Library Script:
   This is an R script file that acts as a library for the functions needed by the main script by pushing them into the workspace. It is run on the beginning of the execution of the main script.
7. R Package Install Script:
   This script is responsible for installing all necessary R packages prior to first run. It is executed by the function library script at the beginning and checks if any of the packages needed are missing and starts the R-integrated download and install process if this is the case.
8. R Workspace File:
   This is a compressed file in which the R workspace is saved every times after running

the script. It contains a selection of R objects including variables, data tables and functions that are deemed to be persistent across multiple script runs as well as for posterior examination.

9. Main Script Output File:
   This is a file in which the test classification table is written by the main script, for the sake of a quick lookup every times after running the script.

10. R Plots:
    This is a PDF file generated by the main script which contains statistical plots that illustrate the outcomes of the calculations and tests performed during main script execution. As plotting is optional in the main script and may be switched off by a parameter at script start, this file may not necessarily be present.

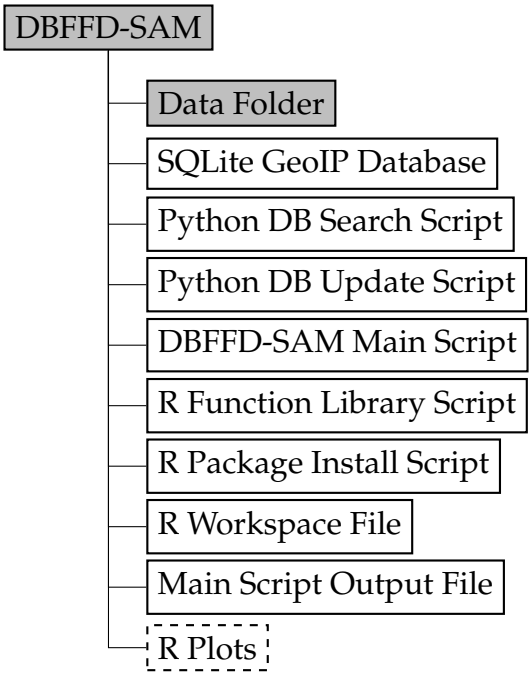

Figure 3.1: Component structure of the implementation

At next it is necessary to explain that the implemented R-script can be started with a number of parameters which determine its operation and thus have an impact on the workflow. As mentioned in section 2.4 the script's parameters are supplied via the –args parameter of the 'R' command. The following table 3.1 gives a summary of all possible script parameters and how they work.

name	is mandatory	type	function
in_file	yes	character	path of input file, basically a list of domain names in CSV format
out_file	yes	character	path to a text file in which the final results of the script will be outputted for a quick lookup
img	yes	character	path to the R workspace image in which objects that are to be persistent between runs will be saved
update_db	no	logical	parameter that controls if the GeoIP database update script should be executed prior to the rest of the script or not; if yes it will cause the database to be updated. If it is left out, a default value of FALSE will be assumed.
randomize	no	integer	specifies if and how many simulated 'malicious' domains with randomized IP locations should be generated. The default value is -1 which means that none of such domains will be created. A value of 0 means that the number of simulated domains will match the number of 'real' domains that can be found in the input file. Any other value will specify the exact number to do so.
plotting	no	logical	specifies whether R plots of the results should be created or not; its default value is FALSE.
no_test	no	logical	specifies if the statistical test workflow should be performed; if set to TRUE the main script will halt before doing so. Can be used to update the database or redownload the R packages needed only; without performing the whole workflow. The default value is FALSE.
no_save	no	logical	this parameters controls whether the R workspace of a script run will be saved on its finish or not. Can be used to prevent the script from saving its results to the workspace at start time for any reason.

Table 3.1: Possible parameters of the implemented R-script

A sample call of the main script can thus look like this (line breaks are for the sake of readability and have to be omitted in practice):

```
R < dbffd-sam.r --args
in_file="data/test_data/domains/malware_domains.csv"
img="test02_2_5.RData" out_file="test02_2_5.out" update_db=TRUE
num_domains=0 randomize=-1 plotting=FALSE
```

The workflow consists of several steps of particular actions and decisions which are going to be outlined here one after another.

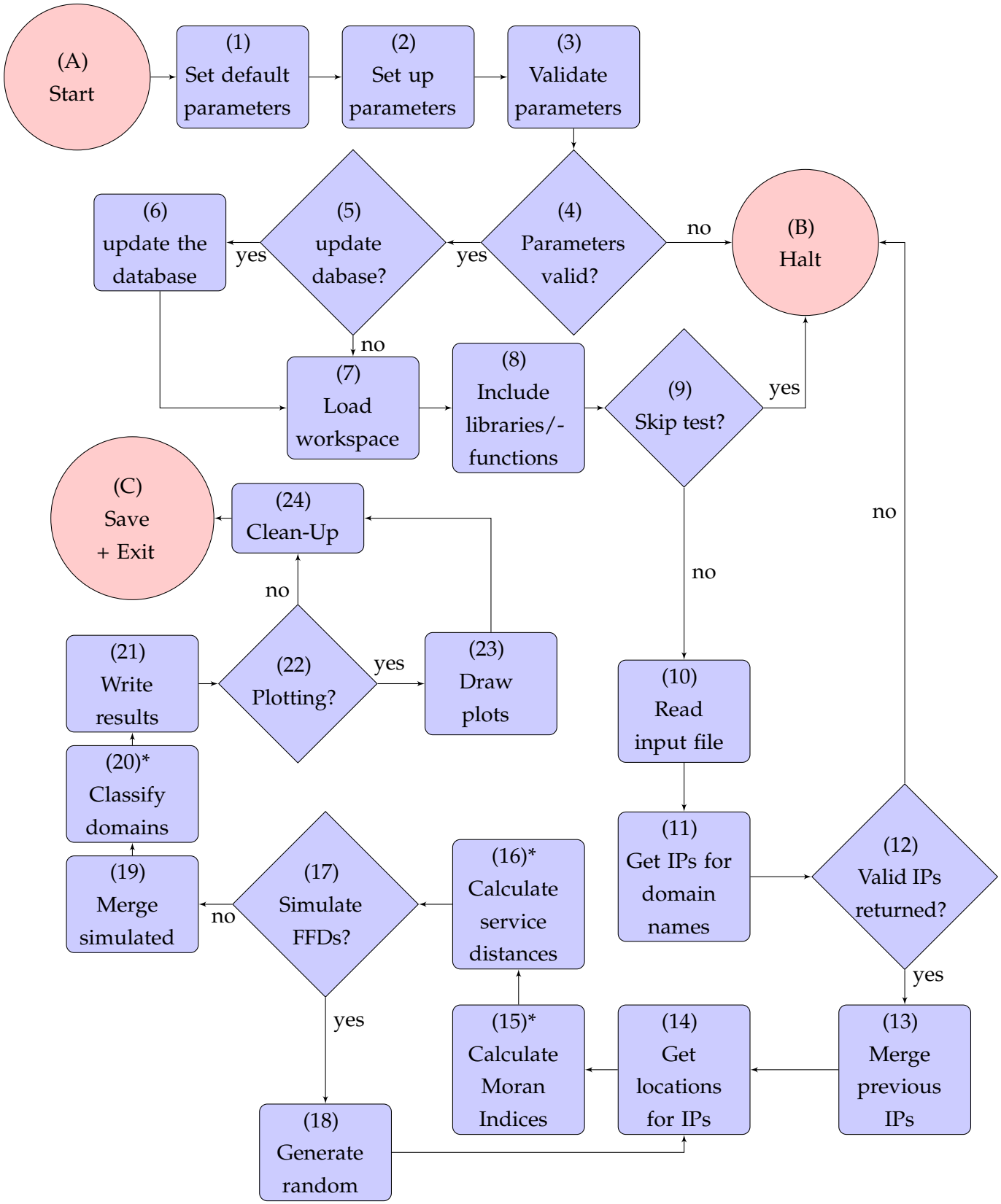

Figure 3.2: Flow-chart of the main workflow

The following describes the different steps of the the workflow, including the start- and stop-points which are designated as (A), (B) and (C). The remainder of the steps is numbered consecutively. A graphical outline of these (as well as the above start- and entry-points) is given as a flow-chart in the above figure 3.2. The steps are marked with an asterisk (*) in the chart are to be furtherly detailed after this.

A Start
This is the main script entry point which is entered on script-start. After this step 1 is executed.

B Halt
This is the premature exit point in which the script will simply stop. It is reached in case of invalid parameters in step 4, if the parameter 'no_test' is set to TRUE after step 9. or if there is another error during execution for any reason.

C Save + Exit
This is the final exit point, that is reached after successfully conducting all tests and other tasks, after the workspace clean-up in step 24.

The following describes the rest of the different steps of the workflow, which are consecutively numbered.

1 Set default parameters
In this step the default values of the non-mandatory parameters of the script are set.

2 Set up parameters
If non-mandatory parameters are supplied by commandline at script start, the default parameters of the previous step are overridden.

3 Validate parameters
In this step all parameters are validated.

4 Parameters valid?
If there were any invalid parameters the script will stop at B.

5 Update Database?
In this, it will be checked if 'update_db' is TRUE. If so, the GeoIP database will be updated in the next step.

6 Update the database
Here the new IP geolocation data is downloaded from Maxmind and integrated into the database as mentioned in section 2.2, if the check in the last step has been successful. This is done by running the corresponding python script for the database update.

7 Load workspace
The R workspace specified in the 'img' parameter is opened and the data-holding objects stored in it are loaded. If the workspace file with the specified name is non-existent, loading the workspace is skipped. The workspace name will be used when saving the necessary objects as it is done in C.

8 Include libraries/functions
The R function library script is run. It will ensure that all necessary packages are installed and all functions are loaded into the current R environment.

9 Skip test?
If the 'no_test' parameter is set to TRUE the script will stop at B.

10 Read input file
The CSV input file which is specified by the parameter 'in_file' is read and its content (the list of domains) is stored as object into the workspace.

11 Get IPs for domain names
In this step the function for getting all IP addresses (Answer and NS-section) for a domain are fetched by using 'dig'. This is achieved by looping through all domain names that have been read from the input file and fetching all A-section IP addresses and all NS-section IP addresses consecutively, just like the sample queries presented in subsection 2.1.1. These IP addresses are then stored alongside their corresponding domain names inside a data object.

12 Valid IPs returned?
Now it is checked if at least one valid A-section and at least one NS-section IP-address have been returned. If not, which could be the case for example if all domains in the input file are non-existent or outdated, the script stops at B.

13 Merge previous IPs
If the previous check was successful, the IPs that have been returned at step 12 are merged with the IP address data objects that have been recovered from the workspace

that have been opened in step 7, if a workspace with the name that was specified in the beginning already existed. If this was not the case, merging the data objects is skipped and the previously returned IP addresses are treated as the complete set.

14 Get locations for IPs
In this step the IP geolocations of the fetched IP addresses are queried from the database by running the database python search script with each respective IP address supplied as parameter. This is done for both, the A-section as well as the NS-section IP addresses. The IP locations for both sections are stored separately in two different data objects alongside with their corresponding domain names for the sake of lucidity when furtherly handling them.

15 Calculate Moran Indices
In this step, the Moran Indices of each domain are calculated by examining their IP locations. This step will be described in more detail within subsection 3.2.3.

16 Calculate service distances
Here, the average service distance of each domain will be calculated, again by looking at their respective IP locations, the proceedings in this step will be also detailed in subsection 3.2.3.

17 Simulate FFDs?
Here the value of the 'randomize' parameter is examined, as described prior in this section. If there are FFDs left to be 'simulated' this will happen in the subsequent step, otherwise the script will jump to step 19.

18 Generate random
In this step simulating an adequate number of Fast-Flux domains is started by generating a respective set of random domain names, each with a corresponding set of 3-5 (this range has been selected for performance reasons) addresses for both A- and NS-section. The non-existent top-level-domain[5] '.bb' is suffixed to random domain names to ensure that no 'real' domains will associated accidentally. As it is stated in section 1.2, the randomized IP addresses however are real ones that are to exhibit a real IP location, although it does not matter which one, merely it is their random spatial distribution that is to let them exhibit a very striking FFD-like behaviour. After generating the IP addresses the script calls the same function as in

5 `http://dictionary.reference.com/browse/top-level+domain` [last access: 14th of June 2013]

step 14 for querying their IP locations and continues through the steps 15 and 16, which is done until the number of FFD to be simulated is reached.

19 Merge Simulated
Here, the randomly generated data of the simulated FFDs is merged with those of the 'real' domains' ones. If there were no FFDs to be simulated, this process is skipped.

20 Classify domains
This is probably the most important step of the script, in which the accumulated domains are classified by as either suspect or non-suspect (see section 1.1). This process is given in detail in the following subsection 3.2.3.

21 Write results
After doing this, the table object that contains the classification results is written to a file (that was specified in 'out_file') in text form, in order to provide a quick lookup of the results.

22 Plotting?
Here it is checked if the 'plotting' parameter is set to TRUE. If this is the case, a certain number of plots are drawn that depict the different results of this script, for example scatter plots and density curves of Moran Index and Average Service Distance values the domains by section and simulated/real type. Examples of these will be presented in chapter 4. If 'plotting' has been set to FALSE the script continues straigth to step 24.

23 Draw plots
Here the actual plots are drawn if the previous test succeeded. Afterwards the script will continue to the next step.

24 Clean-Up
Here all objects are removed from the workspace that are not intended to be saved to the workspace image. Once this is done the script will finish its execution in C, where the remaining objects will be saved to the workspace image file that has been specified in the beginning.

The next subsection tells about how the actual statistic classifiers have been implemented and shows the crucial workflow steps mentioned in the previous description in more detail respectively.

3.2.3 Realisation of statistic classifiers

As mentioned in section 1.2 the two statistic classifiers this approach relies on in order to characterize the spatial behaviour of a particular domain are the Moran Index of spatial autocorrelation and the average service distance as it has been described in section 2.3. In this section the workflow steps 15, 16 and 20 that correspond to calculating said classifiers and carrying out the classification of the domains respectively, are going to be explained in more detail, each one after another.

(15) Calculate Moran Indices
In order to calculate the Moran Indices, a set of A-section IP locations stored by their corresponding domain names is supplied to the responsible function, which then loops through the whole set of domains. For each domain, its IP locations are examined and a distance matrix is calculated by using their latitude and longitude coordinates, which is then used to calculate the corresponding Moran Index straight afterwards. The indices are then stored together with their corresponding domain names. There are two special cases that can occur during this:

- There is only one IP location found for a domain.
  This happens when a domain is associated with only one IP address. In this case calculating the Moran Index will be erroneous as no valid distance matrix could be created.
- A domain has several associated IP addresses, but their IP locations are all in the same place.
  This is for example the case for 'google.com' which has many addresses associated which are all found to reside in one location. Here, the calculation of the Moran Index would also fail because the distances within the distance matrix being all zero.

Both of these cases are handled by assuming a Moran Index of 0. This value normally indicates spatial randomness, however it could be observed during several script runs by examining the results that this value won't naturally occur in this context. The reason for this lies in the nature of the attributes that are used to calculate the distance matrices and the Moran Indices respectively, namely the latitude and longitude coordinates of the different domains' IP locations. In the case of spatial randomness, the attribute values of the observed points in space won't exhibit any clear correlation to their position; in this case, however, the coordinates of the points itself are used as attributes which means that they

must be at all accounts correlated to the position of the points they describe. For this reason a Moran Index of zero has been chosen as an indicator in the case that no valid Moran Index could have been calculated for a particular domain.

(16) Calculate Service Distances
The calculation of the average service distances is straight forward. For each domain the euclidean distance between every A-section IP location and every NS-section IP location that are associated with that domain, is determined and the mean is calculated out of this values. These mean values are then stored alongside their corresponding domain names and returned.

(20) Classify Domains
Here, the actual domain classification as either suspect or non-suspect is performed; this process is straightforward and comparatively simplistic. It is done by looping through all domains and checking their Moran Index and average service distance values against certain threshold values. These threshold values have been determined by repeatedly examining the Moran Index and average service distance values of a list of reportedly legit domains as well as a list of reportedly malicious domains during multiple subsequent test runs of the script. The observations made by investigating the values of the lists of domains have been used to model the threshold criteria in such a way that as least as possible false positives and negatives would be achieved.

In order to understand how this has been done exactly, it is necessary to present some key observations that helped determining the threshold values, which are the following:

- The majority of processed domain names, regardless of whether they are legit or malicious expose just one single IP address or one single IP location respectively, even when observed continuously over certain time span. These have been determined to be as definitely Non-Fast-Flux as this trait contradicts Fast-Flux behaviour per definition.
- Large CDNs like google.com for example, have many IP addresses associated, but these are all located in the same place, which effectively leads to the same result as in the previous point.
- From the set of domains where Fast-Flux behaviour could be precluded, it could be observed that the majority of average service distance values lay within a certain

percentile range, this has been exploited to define a threshold value for the average service distance.

After classifying the domains as either suspect or non-suspect, the steps 21-24 are performed as described in the previous subsection 3.2.2, before the workspace is saved and the script exits.

The results that have been achieved by the aforementioned procedural manner are presented in all detail in the following chapter 4.

3.3 Measuring Accuracy

Measurement accuracy is defined as the "closeness of agreement between a measured quantity value and a true quantity value of a measurand" (JCGM/WG2 2008). In this work accuracy will be thus represented by the rate of the number true positives and negatives to the sum of true positives, true negatives, false positives and false negatives:

$$A = \frac{n_p + n_n}{n_p + n_n + \bar{n}_p + \bar{n}_n}$$

where:

- A is the accuracy value in percent
- n_p is the number of observed true positives
- n_n represents the number of true negatives
- $\bar{n}_p$ is the number of false positives
- n represents the number of false negatives

The accuracy of the outcomes of the statistical tests will be assessed in the next chapter 4 by using this method.

4 Results and Interpretation

In this chapter, the results that have been achieved by the research done during the course of this work will be presented and interpreted.

4.1 Characterization and interpretation

The totality of domain names that have been examined during the tests, is (as already indicated in section 3.1) made up of three different sets of domains: the list of legit domains, the list of malicious domains as well as the simulated FFDs that have been generated by the script. At first, for the legit and malicious domains, a selection of domains will be investigated in detail, in order to present all relevant cases that occurred within the results of the test series; after that overall statistics, for all sets of domains will be explained. Table 4.1 summarizes a selection of statistical values for all three sets of domains; these values comprise the following:

- total domains
  This is the overall number of domains in the set.
- domains classified
  This is the number of domains that made it to the classification towards the end of the script. A difference between the total number of domains and the number of domains classified can be caused by domains that are unreachable, unresolvable or do not yield any valid IP location for any reason; those will be excluded from the set of domains that reach the classification procedure.
- positives
  This is the number of domains that were classified as 'suspect'.

- negatives
  This is the number of domains that were classified as 'non-suspect'
- true positives
  This is the number of domains that were classified correctly, as 'suspect'.
- false positives
  This is the number of domains that were classified erroneously, as 'suspect'.
- true negatives
  This is the number of domains that were classified correctly, as 'non-suspect'.
- false negatives
  This is the number of domains that were classified erroneously, as 'non-suspect'.
- accuracy (%)
  This is the classification accuracy which is given by the formula in section 3.3.
- service distance 95% percentile
  This is the 95% percentile of the average service distance values of the domains within the set that have been precluded to be Fast-Flux domains; this means that 95 percent of these domains have an average service distance that lies below this value.

value	legit	malicious	simulated
total domains	501	832	500
domains classified	489	541	428
positives	4	4	428
negatives	485	537	0
true positives	0	4	428
false positives	4	0	0
true negatives	485	530	0
false negatives	0	7	0
accuracy (%)	99.182	98.7061	100
service distance 95% percentile	7997.547	7889.459093	10583.076

Table 4.1: Overall statistics for the different sets of domains

As there is no clear 'resolution' available whether the results of the classification are correct or not, the following assumptions had been inferred from the background of

the used data, when it came to handling true or false positives and negatives respectively:

- Legit domains:
  As all of the domains in this list are trusted, every domain that is classified as 'suspect' will count as a false positive.
- Malicious domains and simulated domains:
  As all domains at the list of malicious domain originate from a bot-net blocklist and are generally distrusted, every domain classified as 'suspect' will be assumed being a true positive with no false positives present at all. In addition to that all domains with a Moran Index of zero will be counted as true negatives, whilst all domains with a Moran Index other than zero that are classified as 'non-suspect' will be deemed to be false negatives.

The next section details the different behavioural traits encountered, by giving examples and statistics for the different sets of domains.

4.2 Behaviour of domains

The key cases that have been encountered (two of them have been already mentioned in subsection 3.2.3) when examining the test results will be presented in detail here, in order to show how different domains behave. This will be done using the examples of some selected domains; their statistical values are summarized in table 4.2.

The list of legit domains comprised well-known sites, with a number of large CDNs among them, such as 'google.com', 'facebook.com' or 'amazon.com'. The first example that will be presented here is a large and well-known content distribution network, the domain 'youtube.com'. This domain represents the case of having many IP addresses associated with, as many as 22, although having only one unique IP location for all of these IPs. In case of youtube.com this is Mountainview, California, the locations were most of google-related services and sites (with 'google.com' being the best example) will have its IP locations. Also, the average service distance proofed to be exactly zero for this, domain; which has its reason in all corresponding name servers being concentrated in the same single IP location; analogously this is also the case for 'google.com'. For this reason the Moran Index of 'youtube.com' also has an assumed value of zero, as just one

unique IP location was found for this domain, which defies the calculation of a distance matrix.

The next example being presented is the domain 'wordpress.com'; it has 6 IP addresses associated with three unique IP locations belonging to them. The Moran Index of this domain is -0.13338235, which indicates moderate dispersion among its IP locations; the average service distance is relatively low with a value of 1270.804. The domain 'wordpress.com' is a border case, which would have been erroneously classified as 'suspect' without the average service distance as second classifier.

Another example from the list of legit domains 'icq.com'. This domain is characterized by having just one IP and one IP location respectively associated (and thus a Moran Index of zero), but an extraordinarily high average service distance of 14357.02. Although this domain defies being classified as 'suspect', it shows that a high average service distance is not necessarily only a characteristic of domains that have many IP addresses associated as well; this case is not quite common however, with only ten domains out of 489 having a Moran Index of zero and an average service distance greater than ten thousand.

The last example from the list of legit domains is 'amazon.de' which is an example of a domain that has been erroneously classified as 'suspect'. It has a Moran Index of -1 with three IP addresses associated that belong to two unique IP locations. The domain has a high average service distance of 13364.257. It is one of the four reported false positives within the list of legit domains, out of a total of 489 classified domains.

From the set of malicious domains only one example will be considered here in detail, which is a domain that has been classified as 'suspect'. As the said example, the domain 'aesssbacktrack.pl' will be taken. It has eight IP addresses associated which correspond to eight unique IP locations. It features a Moran Index of 0.012668304 and an average service distance of 9507.62. The spatial behaviour of this domain exactly matches the characteristics that are expected from a bot-net related Fast-Flux domain as it has been described in section 2.3 towards the beginning of this work.
The next example is from the set of simulated Fast-Flux domains. This set is special inasmuch as the domains in this set have been generated randomly by selecting associating arbitrary to random domain names. This has to be done to simulate a set of domains that adhere to the most explicit spatial behaviour that is expected from a Fast-Flux domain in theory. A typical example of the domains in this set would be 'Hubbart Jr.bb' which

is characterized by a Moran's Index of -0.0487104772 and an average service distance of 10280.711. Obviously this domain has been, as intended, classified as 'suspect'. The reason why the number of IP addresses and unique IP locations are not given for this example, is that during the simulation process the two classifier values are calculated on the fly by randomly creating IPs, which are the discarded and not stored, for not wasting working memory.

At last, for the sake of completeness, an example has to be given which is representative for the majority of domains that have been processed and classified; a domain that has just one associated IP address and thus one associated IP location. This domain necessarily has a Moran Index of zero as well as an average service distance of zero. An example of such a domain (taken from the list of legit domains) would be 'blogspot.com'.

domain	IP addresses	IP locations	Moran Index	service distance	suspected
youtube.com	22	1	0	0	no
wordpress.com	6	3	-0.13338235	1270.804	no
icq.com	1	1	0	14357.02	no
amazon.de	3	2	-1	13364.257	yes
aesssbacktrack.pl	8	8	0.012668304	9507.62	yes
Hubbart Jr.bb	-	-	-0.0487104772	10280.711	yes
blogspot.com	1	1	0	0	no

Table 4.2: Examples of different domain characteristics

For comparison, figure 4.1 illustrates plotted density curves of both statistic classifiers, for all three sets of domains.

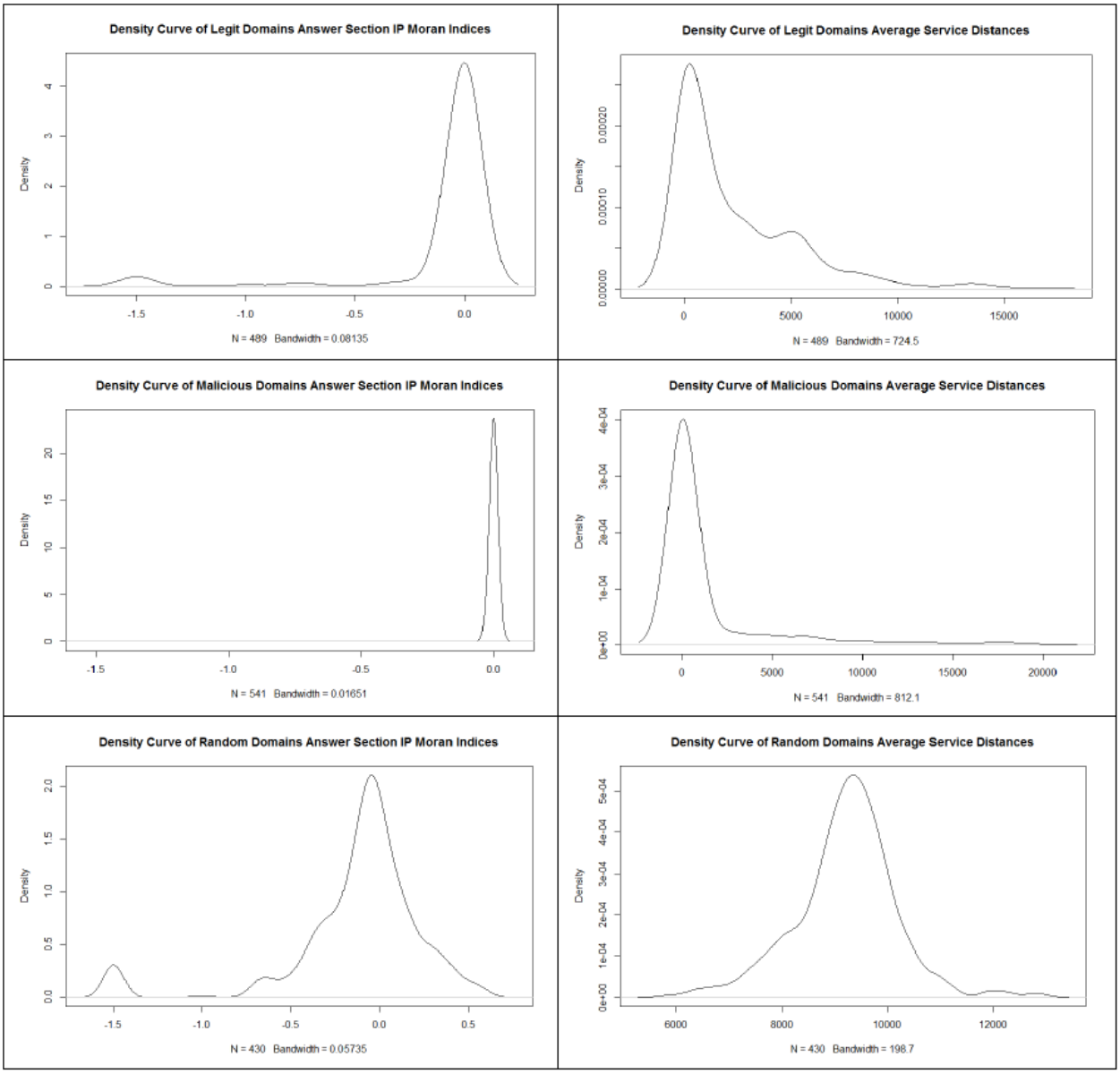

Figure 4.1: Comprehensive density curves of statistic classifiers

Finally, the following figure 4.2 visualizes the spatial distribution of the said domains on a world map (compare figure 2.2 in subsection 2.1.2).

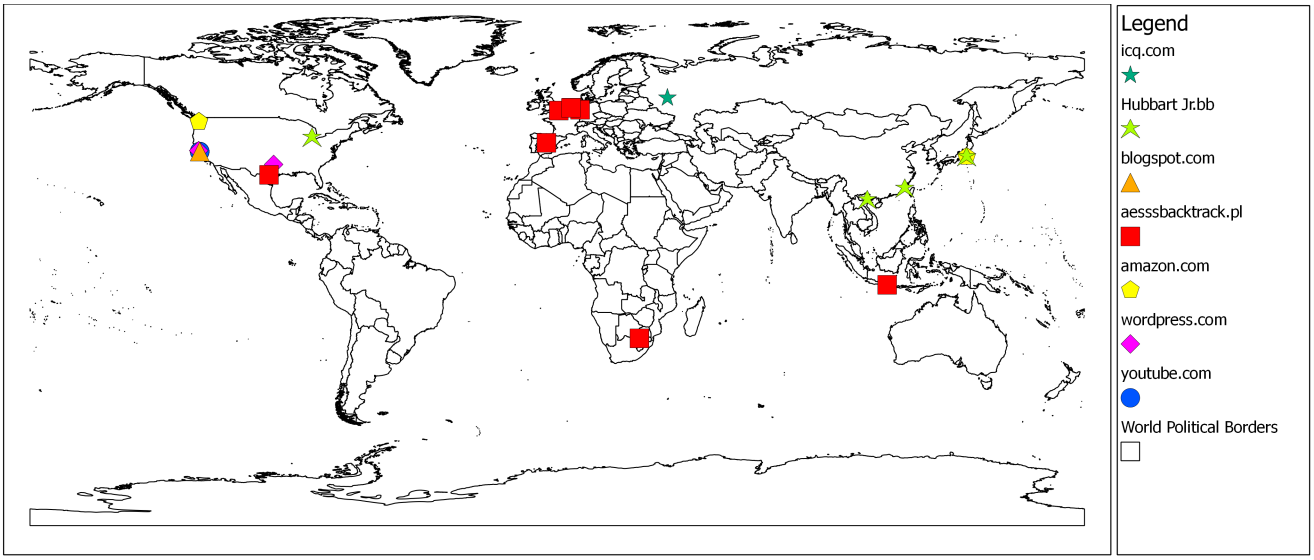

Figure 4.2: Spatial distribution of selected domains

4.3 Statistic outcomes

After the encountered key cases have been presented, the overall statistics for the three different sets themselves are going to be discussed in straight forward manner. The statistical values of these are summarized in table 4.1. The first set that will be dealt with is the set of legit domains. Of 501 domains that have been read from the input file a total of 489 have been classified, with four positives (counted as false positives) and 485 negatives (counted as true negatives), resulting in a classification accuracy of 99.182 percent. 95 percent of the average service distance values of the domains with a Moran Index of zero lay below a value of 7997.547. This means that all domains with non-zero Moran Index that have an average service distance that lies above this threshold are classified as 'suspect'. Here, the average service distance classifier helped greatly to reduce the false positives, a total of 45 domains would have been classified as 'suspect' if the Moran Indices had been examined only, reducing the classification accuracy to 91 percent.

In comparison to that, the statistical values of the set of malicious domains can be examined. Here, a total of 832 domains have been given as input, whereas only 541 of them have been classified. The reasons for this lies in the fact, that many addresses of the blocklist that were

used as data source, could not be contacted or resolved at the time of testing, although the said blocklist is maintained and updated continuously. Thus it seems that domains that are involved in bot-net activities do have a short lifetime and are frequently disabled, either by authorities or the operator of the bot-net himself. Of these 541 classified domains a total of four has been marked as 'suspect', which are counted as true positives, in contrast to 537 negatives, from which seven domains have a Moran Index not equal to zero. The latter ones have thus been counted as false negatives, resulting an overall classification accuracy of 98.7061 percent. All of these seven false positives have been classified in that way because of an average service distance that lay below the 95% percentile threshold value of 7889.459093. That this value is lower then the one observed in the set legit domains, again indicates that the correlation between the average service distance and the nature of a domain is not as strong as expected, at least when dealing with real-world data.

This can be clarified when examining the statistic properties of the set of simulated domains, that has an average service distance threshold value of 10583.076, which is significantly larger than the one of the former two sets of domains. As the IP addresses for each simulated domain comprises of a set of randomly selected IP addresses greater than one, single IP locations are extremely unlikely, which explains that no Moran Index with a value of zero can be found in this set of domains. The spatially arbitrary selection of IP addresses also explains the large average service distance percentile observed. From this set of domains 100 percent have been classified as 'suspect' as it was intended by generating the set, which shows that the spatial classifiers are working very well for theoretical data, but not as same as well for non-simulated real-world data, as it can be observed by examining the results for the former two sets of domains. It has to be noted that from the total of 500 simulated domains only 428 have made it to classification. This can be explained by the fact that the randomly generated IP addresses often hit addresses which are reserved on the internet for special purposes, and thus cannot be associated with a valid IP location.

Further, it can be said that the two spatial statistic classifiers that were used in order to realise a classification of the different domains as either 'suspect' or 'non-suspect', thus the Moran Index of spatial autocorrelation and the average service distance of nameservers, in a large part, followed the behaviour that was proposed in section 2.3; with some meanderings however. Viewed separately the Moran Index showed no apparent unexpected behaviour at all, being able to indicate spatial dispersion for a domain's IP location, as

well as leaving CDNs unsuspected, that have their IP addresses organized in a central location. The average service however, on the one side, in combination with the Moran Index, helped to reduce false positives among the legit set of domains, but potentially increased false negatives on the other side, when going over the set of malicious domains. Nevertheless, in 57 cases were the average service distance was deciding, it acted right in a total of 41 cases which gives a success rate of 77.19298 percent, which credits the average service distance being a less performing classifier than the Moran Index, but still achieving correct results when being combined with the former classifier. The lacking availability to differentiate between domains with just one and more than one unique IP locations however, means that it is inalienably dependable on the Moran Index to perform.

The next chapter concludes this work and gives a statement of the overall quality of the results achieved as well of the validity of the methods applied.

5 Conclusion

The research that was conducted in the course of this work, has shown that the combination of the two statistic classifiers proposed, provides an accurate and lightweight possibility to detect Fast-Flux behaviour among a set of given domains; while avoiding false positives among legit Content Distribution Networks to a certain degree. It should be kept in mind, that the setup approached during this work, has been realised with a comparatively low expense of resources, the only commercial component being the virtual server on which the final setup has been implemented.
Neither traffic-intensive network monitoring nor any commercial data sources have been employed here, the main data traffic that was present here, was due to frequent DNS queries and occasional updates of the FreeGeoIP database, which is, at least for the former, negligible.
Certainly, this approach in terms of data processing, data gathering and how threshold values are determined can be optimized, especially when interpreting the average service distance values, which means that there may be more accurate methods of classifying a domain according to the average service distance than just using a 95% percentile. One way to improve the classification accuracy, that does not apply solely to the average service distance, but to all classifiers however, may be incorporating a Bayesian learning network[1], which is basically a learning probabilistic model, which could be continuously modified on a trial-and-error basis, in order to achieve the best possible results. As well, other classifiers could be incorporated in this process to augment accuracy and confidence in classification, such as the time to live for the IP addresses for a certain domain, which could help to more reliably distinguish malicious from legit Fast-Flux domains, as the former exhibit a characteristically low time to live value for their IPs, which can be read in subsection 2.1.2.
The observations made during conducting the research also indicate that the quality of the outcomes that can be achieved by these means, heavily depends on the input, thus the sets of domains investigated, being as up-to-date as possible; as Fast-Flux domains were

1 `http://www.ee.columbia.edu/~vittorio/Lecture12.pdf` [last access: 14th of June 2013]

observed to expose a short lifetime only. Due to this, it would be interesting to see this setup to be operated in the context of a professional cyber-defence project, were live-data is collected and evaluated continuously, such as the ACDC centralized data clearing house, in order to see how this approach performs in such an environment.
As a final word, it has to be said that though Fast-Flux detection by means of spatial statistics seem trivial, especially when compared to other approaches, decent maintenance and apprehension needs to be provided in order to operate these; input as well as output constantly needs to be interpreted in a critical and qualitative way, which basically means that these always need to be double-checked and handled with common sense, in order to prevent the classification process from yielding erroneous results.

List of Figures

List of Tables

Bibliography

References

abuse.ch: 2013, zeustracker.abuse.ch, `https://zeustracker.abuse.ch/blocklist.php`. [Online; accessed 30-April-2013].

Alexandre Fiori: 2013, freegeoip.net, `http://freegeoip.net/static/`. [Online; accessed 19-March-2013].

Antonio Gasparrini: 2011, R software: advantages and opportunities, `http://csm.lshtm.ac.uk/files/2011/10/Antonio-14-10-2011.pdf`. [Online; accessed 30-April-2013].

Caglayan, A., Toothake, M., Drapeau, D., Burke, D. and Eaton, G.: 2009, Real-time detection of fast flux service networks, *Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications and Technology* .

Feily, M., Shahrestani, A. and Ramadass, S.: 2009, A survey of botnet and botnet detection, *Emerging Security Information, Systems and Technologies, 2009. SECURWARE '09. Third International Conference on* .

Gill, P., Ganjali, Y., Wong, B. and Lie, D.: 2010, Dude, where's that ip? circumventing measurement-based ip geolocation, *Usenix Security Symposium* .

Griffith, D. A.: 2009, Spatial autocorrelation.

IBM: 2008, *System i: Networking Domain Name System, Version 6 Release 1*, IBM.

JCGM/WG2: 2008, *International vocabulary of metrology — Basic and general concepts and associated terms (VIM)*, JCGM.

linux.die.net: 2013, dig(1) - Linux man page, `http://linux.die.net/man/1/dig`. [Online; accessed 19-March-2013].

malwaredomains.com: 2013, malwaredomains.com, `http://www.malwaredomains.com/`. [online; accessed 14-June-2013].

Nair, H. S. and Ewards, V.: 2012, A study on botnet detection techniques, *International Journal of Scientific and Research Publications* .

R Foundation: 2013, www.r-project.org, `http://www.r-project.org//`. [Online; accessed 19-March-2013].

seomoz.org: 2013, seomoz.org top 500 domains, `http://www.seomoz.org/top500`. [Online; accessed 19-March-2013].

Stalmans, E., Hunter, S. O. and Irwin, B.: 2012, Geo-spatial autocorrelation as a metric for the detection of fast-flux botnet domains, *Information Security for South Africa (ISSA)* .

Venebles, W. N. and Smith, D. M.: 2013, *An Introduction to R*, R Foundation.

Wang, H.-T., Mao, C.-H., Wu, K.-P. and ming Lee, H.: 2012, Real-time fast-flux identification via localized spatial geolocation detection, *Computer Software and Applications Conference (COMPSAC)* .

Zhang, L., Yu, S., Wu, D. and Waters, P.: 2011, A survey on latest botnet attack and defense, *International Joint Conference of IEEE TrustCom-11/IEEE ICESS-11/FCST-11* .

Zhao, D. and Traore, I.: 2012, P2p botnet detection through malicious fast flux network identification, *Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing* .