

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering

DIPLOMA THESIS ASSIGNMENT

Student: **Bc. Vojtěch Létal**

Study programme: Open Informatics
Specialisation: Artificial Intelligence

Title of Diploma Thesis: **Discovering of malicious domains using WHOIS database**

Guidelines:

Search and study the prior art on detection of malicious domains.
Analyse the available WHOIS and NetFlow data.
Design an algorithm for discovering yet unseen malicious domains using the available data.
Evaluation of the implemented algorithm.

Bibliography/Sources:

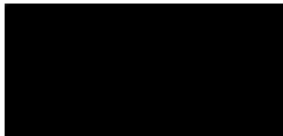
D. Koller and N. Friedman - Probabilistic Graphical Models: Principles and Techniques.
edited by - MIT Press (2009)

Lancichinetti, A. and Fortunato S. - Community detection algorithms: a comparative analysis
- Physical review E 80.5 (2009)

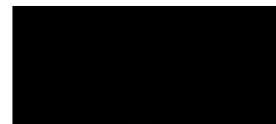
White, T. - Hadoop: The Definitive Guide (1st ed.) - O'Reilly Media, Inc. (2009)

Diploma Thesis Supervisor: Ing. Tomáš Pevný, Ph.D.

Valid until the end of the summer semester of academic year 2015/2016



doc. Ing. Filip Železný, Ph.D.
Head of Department



prof. Ing. Pavel Ripka, CSc.
Dean

Prague, March 26, 2015

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering



Master's Thesis

Discovering of malicious domains using WHOIS database

Bc. Vojtech Letal

Supervisor: Ing. Tomas Pevny, Ph.D.

Study Programme: Open Informatics

Field of Study: Artificial Intelligence

May 11, 2015

Acknowledgements

First I would like to thank to my supervisor Tomáš Pevný for his guidance and immense patience and to Václav Šmídl for his advices about the Variational Bayes approach and Bayesian inference in general. My deep gratitude belongs to my family and to my girlfriend Adéla for the love and enormous support during my studies. And at last I would like to thank to all my past and present teachers and professors, some of which were really extraordinary, for providing me with all the necessary knowledge and not giving up on me.

Declaration

I declare that I elaborated this thesis on my own and that I mentioned all the information sources and literature that have been used in accordance with the Guideline for adhering to ethical principles in the course of elaborating an academic final thesis.

In Prague on May 11, 2015

.....

Abstract

The internet infrastructure is heavily abused to deliver malware and to control infected computers. Blacklisting techniques are used to prevent the user from being infected while surfing the web. The problem of those techniques is that a domain must be detected to actually serve a malicious purpose before it is blacklisted. In this thesis we propose a novel reputation based system which is able to give an estimate of maliciousness even for domains which it have not yet observed. To build the prior estimate the system utilizes information extracted from WHOIS records.

Abstrakt

Internet je zneužíván jako médium pro šíření virů a ovládnutí již infikovaných počítačů. Techniky využívající blacklisty se snaží ochránit uživatele procházející internet od potenciálních infekcí. Problem těchto technik je, že doména se objeví na blacklistu až poté, co je zaznamenána nějaká zákeřná aktivita s ní spojená. V této práci představujeme reputační systém, jenž je schopen dát odhad zákeřnosti domény i v případě, že jsme ještě nezaznamenali žádnou aktivitu s ní spojenou. Systém k tomu využívá informace získané z WHOIS záznamů.

Contents

1	Introduction	1
1.1	Motivation	1
1.1.1	Used notation	4
2	Prior art	5
2.1	DNS analysis	5
2.1.1	EXPOSURE	5
2.1.2	Notos	6
2.1.3	Probabilistic Threat Propagation	6
2.2	WHOIS analysis	7
2.2.1	We know it before you do	7
2.2.2	Predictive domain blacklisting	8
2.2.3	Meta-data driven classification	9
2.3	Reputation system based on observed evidence	9
3	Proposed Solution	11
3.1	Proposed model	12
3.2	Inference using Variational Bayes	17
3.2.1	Marginals of the m_d	18
3.2.2	Marginals of the a_k	19
3.2.3	Marginals of the b_k	20
3.2.4	Approximation of the normalizing coefficient	21
3.3	Algorithm in steps	22
3.3.1	Initialization strategy	22
3.3.2	Convergence criteria	23
3.4	Implementation	23
3.4.1	Processing WHOIS records	24
3.4.2	Compact graph structure	24
3.4.3	Parallelization	25
3.4.4	Problems with the approximation	25
3.4.5	Handling missing keys	25
4	Evaluation	27
4.1	Empirical analysis of convergence	27
4.2	Datasets	28

4.3	Measures used to evaluate the performance	30
4.4	Discussion of meaning of the parameters Θ	32
4.5	Optimization of the parameters	33
4.6	Comparison with the Probabilistic Threat Propagation	33
4.7	Analysis of the inferred values	35
4.8	Overall performance	36
5	Conclusion	39
A	Variational Bayes method	43
A.1	Factorized distributions	44
B	Nomenclature	45
C	CD content	47

List of Figures

1.1	Malware samples analyzed by AV-TEST p.a.	1
1.2	Spam message from String of Pearls	2
1.3	The credentials used in String of Paerls	3
1.4	WHOIS query result example	3
3.1	Bipartite graph of keys and domains	12
3.2	Histogram of number of observed connections per domain	13
3.3	Graphical representation of the tensor extrapolation.	14
3.4	Dependency of the variables	17
3.5	Supplying missing keys	26
4.1	Example of convergence of the proposed algorithm	28
4.2	Convergence of the Beta prior	29
4.3	Training and testing data explained using Venn Diagram	30
4.4	Optimization of the parameters Θ	34
4.5	Comparison of the proposed algorithm with PTP	35
4.6	Evaluated performance metrics	36
4.7	Correlation between # of connections and error in inference	37
4.8	Overall performance of the classifier based on m_d	38
C.1	Content of the attached CD	47

List of Tables

3.1	Number of keys per domain	25
4.1	Initial parameters of the algorithm	27
4.2	Statistics of the available datasets	29
4.3	Relative occurrences of TLDs in the Sophos blacklist	30
4.4	Definitions of outcomes of hypothesis testing	31
4.5	Basic measures used to evaluate binary classifiers	31
4.6	The chosen parameters of the model Θ	33
4.7	Optimal operating point according to the F-Measure	37

Chapter 1

Introduction

Malicious software, often shortened to malware, is nowadays a serious threat. People do not realize how vulnerable their computer is connected to the Internet. With the rapidly rising number of different technologies we use and their complexity, no-one can be aware of all their vulnerabilities. For example with the current popularity of smart-phones a new generation of malicious software targeting this platform appeared. Using this software attackers are able to intercept SMS communication of victims to get, for example, confirmation codes for banking transactions [4],[1]. This might be a rare example, but as you can see in Fig. 1.1 the number threats on the Internet is growing exponentially over time. It is therefore important to deeply focus on computer and network security.

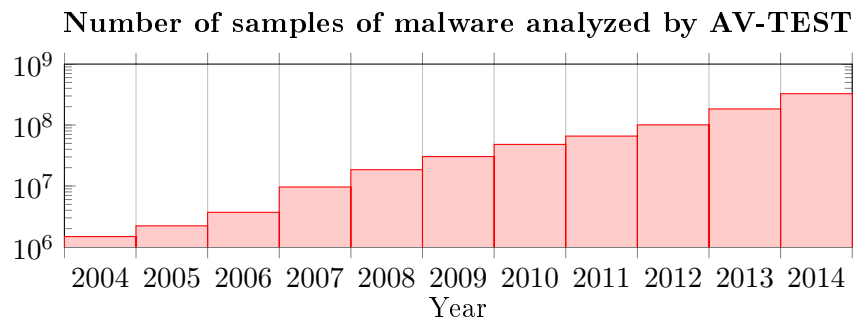


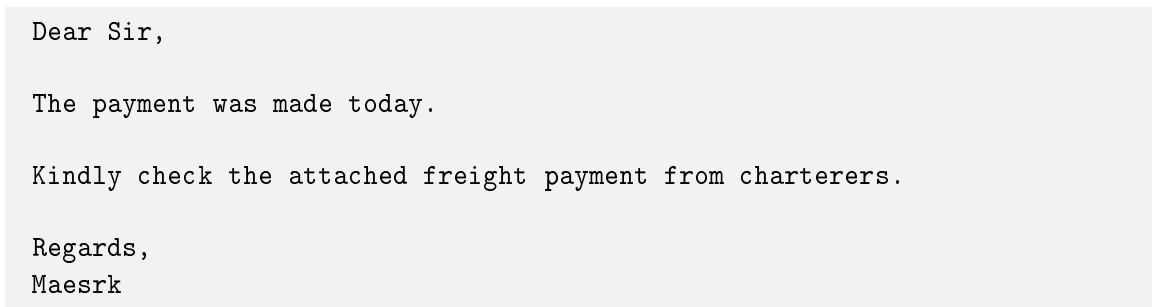
Figure 1.1: The count of malware samples analyzed by AV-TEST institute since 2003. The y-axis has logarithmic scale. You can see that the number of analyzed samples grows exponentially.

1.1 Motivation

The days when viruses were written by programmes to simply show off their skill are now gone. Malicious software is nowadays created for the sole purpose of monetization. After a successful infection malware usually resides in the infected computer waiting for commands

from its master. When activated it might be used for example to send spam, be part of denial of service attacks, mine Bitcoins, to deploy another malware to the infected host, etc.

To be able to keep the population of infected hosts viable by spreading the malware and to control the already infected hosts, the malicious actors need an infrastructure of internet domains and servers. During the summer of 2014, Craig Williams from TALOS posted a series of two interesting articles on their blog [3, 2] about unrevealing of such structure.



```
Dear Sir,  
  
The payment was made today.  
  
Kindly check the attached freight payment from charterers.  
  
Regards,  
Maesrk
```

Figure 1.2: A spam message which was used to deliver the infected document.

The investigation started with a Microsoft Word document `2014-05.doc`, which contained malicious Visual Basic script. The file was distributed by spam in the form of an invoice, purchase order, or receipt, written specifically for the recipient (see Fig. 1.2). After executing the script contacted few different domains on the Internet to download and deploy an executable binary of malware. Those domains were

- `dl.dropboxusercontent.com`
- `londonpaerl.co.uk`
- `selombiznet.in`

The first domains is simply a download server of Dropbox, but the other two domains were interesting. They were registered during the same day, by the same organization, and using similar email addresses. Also there was a suspicious pattern in the postal address used by the registrant. Both addresses contained phrase `number 2 close off medical road`. Other previously unknown domains with suspiciously similar WHOIS information were found by searching for this string in WHOIS records.

By thorough analysis, the operatives from Talos were able to reveal the structure of the malicious campaign. It consisted of 23 different domains which were used to deliver several different types of malware. As you can see in Figure 1.3, 5 different email addresses and 2 different organization names were used to registering the domains. This example shows how valuable the WHOIS records might be for identifying of possibly malicious domains.

The WHOIS protocol, which provides the records, was defined in RFC 3912 [11]. In its early years the protocol was simply used as a directory service. Nowadays the spectrum of information which it provides is much broader including registrant, technical, billing, and administrative contact information.

Email addresses:	Organization names:
<ul style="list-style-type: none"> • selom70@gmail.com • mobday70@gmail.com • davieesselonet@info.ee • wamglu795@126.com 	<ul style="list-style-type: none"> • Adadans Ltd • MediaServicePlus Ltd

Figure 1.3: The unique credentials used by the malicious actor to register the domains.

Domain name: londonpaerl.co.uk	Registrar: PDR Ltd.
Registrant: MediaServicePlus Ltd.	Relevant dates: Registered on: 21-Mar-2014 Expiry date: 21-Mar-2016 Last updated: 08-Apr-2015
Registrant type: Unknown	Registration status: Registered until expiry date.
Registrant's address: 2 close medicle road london Bexley DA5 1ND United Kingdom	Name servers: ns1.suspended-domain.com ns2.suspended-domain.com

Figure 1.4: Example of a result of a WHOIS query for a domain londonpaerl.co.uk

An example of query is shown in Figure 1.4. From the historical reasons the protocol provides results to queries results in a human readable form, lacking any formating specification [5]. This causes that even individual registrars does not guarantee to have consistent formating for the records which they provide. This makes automation of processing of the WHOIS records really hard. Custom parsers are needed to parse the records, but parsed values are not always correct, which brings a lot of noise into the data.

For the methodology and maintenance of the gathering of registrant information is responsible Internet Corporation for Assigned Names and Numbers (ICANN). According to this non-profit organization the registrars are responsible for validity of the WHOIS records they provide. Despite this fact a lot of WHOIS records is now being anonymized. There are services on the Internet which offer registering domains on their behalf, so that the true registrants remain private. Those services are widely used not only to cover malicious activity, but also by individuals which do not want to expose their contact information in the publicly available WHOIS records. It is because the email and even the postal address is often used to deliver fake bills, contracts, and spam to the registrants [14].

As you can see the information in WHOIS records is quite coarse-grained. However, as was demonstrated by Craig Williams, the WHOIS records can be used to find previously unknown malicious domains by exposing relationships between otherwise unrelated domains. Our goal is to design a reputation system for domains which would utilize this property to get a prior estimate about maliciousness of domains if little or no evidence about their behaviour is available. The prior should be build upon information about relationships of domains extracted from the WHOIS records and actually observed evidence.

1.1.1 Used notation

Before we start analysing the prior art we have to introduce an important notation. Let us have two functions $f(x)$, and $g(x)$. The operator *proportional to*

$$f(x) \propto g(x) \tag{1.1}$$

is usually used in the sense that there exists a constant c such that

$$f(x) = c g(x) \tag{1.2}$$

In the case when the $f(x)$ and $g(x)$ describe probability distributions we often omit the coefficient c and use the notation with \propto . The normalizing constant might have very complex form and it is therefore more convenient to work without it. In the end the, the c can be obtained by normalization of $g(x)$ as

$$c = \int_{x \in X} g(x) dx \tag{1.3}$$

and then $f(x) = \frac{g(x)}{c}$ is a valid probability distribution.

In the following text we mostly work with logarithms of probability distributions. We therefore reuse the operator \propto as

$$\log f(x) \propto \log g(x) \tag{1.4}$$

in a sense that there exist a constant c' such that

$$\log f(x) = c' + \log g(x) \tag{1.5}$$

where $c' = \log c$. The form is different, but the meaning of the operator \propto , with respect to the probability distributions, remains the same. This allows us to continually omit coefficients which are constants with respect to x , because we can normalize the $g(x)$ in the end. You should be able to easily distinguish between the two cases from the context of the text.

Chapter 2

Prior art

Malicious actors use the Internet infrastructure to infect computers and to remotely control the infected computers. Because of the low accuracy of standard techniques involving static analysis of malware [6], such as anti-virus solutions, detecting of domains possibly involved in malicious activity is becoming increasingly more important. The prior art described below address this problem and include analysis of passive DNS data and WHOIS records to reveal indirect relationships between domains, as for example shared IP address, and to extract features for training classifiers. In this section we will review some of the articles which are relevant to this thesis.

2.1 DNS analysis

Analysis of passive DNS data has been thoroughly studied recently. It contains fine-grained information about mapping between domain names and IP addresses of servers. This data can be used to find relationships between otherwise unrelated domains and used to expose previously unknown malicious domains based. Their usage might be similar to the WHOIS records and we should therefore focus on studying results achieved in this area.

2.1.1 EXPOSURE

Bilge et al. have proposed a system called EXPOSURE, which is designed to classify domains to malicious / benign [7]. The system uses a decision tree algorithm trained on features extracted from historical DNS data. Those features include

1. **Time-Based Features** daily similarity of the queries, repeating patterns etc.
2. **Answer-Based Features** number of distinct IP addresses, number of domains sharing single IP etc.
3. **Time-To-Live-Based Features** average TTL, standard deviation of TTL etc.
4. **Domain-Name-Based Features** longest meaningful substring, numerical characters etc.

To get a ground truth they collect publicly available blacklists and whitelists and use it to train the classifier. The whole solution is prepared to be used in an on-line setup, continually collecting passive DNS data and retraining the classifier every day. When tested in off-line mode on heavily pre-filtered data the authors claim that the system to has a 98% true positive rate and around 1% of false positive rate.

This approach relies on the way how the DNS is abused by malicious actors to hide their infrastructure. It might be therefore hard to evade such technique, because without this effort the malicious networks would be much more liable to be taken down. The major limitation is that at least 20 observed resolved DNS queries are needed to extract the features for a single domain.

2.1.2 Notos

Concurrently to Bilge, Antonakakis et al. proposed a system called Notos, which is also supposed to find malicious domains based on analysis of DNS data. Although they collect the data in a similar way as Bilge, their approach differs. The features they use are much more complex, depending on DNS zones and IP addresses associated with different domain. The main three categories of features were

1. **Network-Based Features** statistics about relationships between domains and IPs
2. **Zone-Based Features** statistics about other domains sharing the same DNS zone with d
3. **Evidence-Based Features** collected data from sand-boxed malware captures and blacklists

The system has an off-line learning mode and on-line classifying mode. The learning mode is quiet complicated. It relies on clustering of the domains based on the features mentioned above and deriving new nontrivial feature vectors from them.

Although they showed that using the clustering the system was able to identify machines infected by the Zeus botnet, in the terms of performance they perform similarly to the Bilge's approach.

2.1.3 Probabilistic Threat Propagation

Carter et al. proposed a general algorithm for estimating a probability that a given entity is malicious given its relationships with other entities and demonstrated it on passive DNS data [9]. To compute the probabilities a graph $G = (V, E)$ needs to be constructed. Vertices of the graph V are the entities, for example domains and IP addresses of servers, and there is an edge $(v_1, v_2) \in E$ if the two entities v_1 and v_2 are related. Also the algorithm requires the relationships to be quantifiable, i.e. it requires to assign weights to the edges.

The resulting values are interpreted as probabilities, so they must be kept in the range $[0, 1]$. To ensure that, it must hold for all $i \in V$, that $\sum_{j \in V} w_{i,j} = 1$.

A set of known malicious entities T , i.e. tips, is needed to initialize the algorithm. For all tips $t \in T$ the value $p(t; G)$ is given in advance and fixed. The probability is then defined as

$$p(x_i; G) = \sum_{j \in N(x_i)} w_{ij} p(x_j; G - x_i), \quad (2.1)$$

where the $G - x_i$ is a graph G without the vertex x_i and its adjacent edges. You can see that this equation has a recursive structure. The vertices are being removed from the graph so they can not contribute to their own values, the authors call this phenomenon a *direct feedback*. Because of the number of combinations which needs to be evaluated grows exponentially with the number of vertices, this equation can not be efficiently computed on a larger graph.

To solve this problem the calculation is approximated using iterative algorithm as

$$p_k(x_i) = \sum_{j \in N(x_i)} C_k(i, j) = \sum_{j \in N(x_i)} w_{ij} (p_{k-1}(x_j) - C_{k-1}(j, i)), \quad (2.2)$$

where the $C_{k-1}(j, i)$ is the portion of $P_{k-1}(x_j)$ which was directly obtained from x_i .

The algorithm mainly focuses on learning from data with only positive class labels, because the tips are supposed to be known malicious entities and the assigned probability $p(t; G)$ should require our confidence in the maliciousness. The authors themselves interpret the values as a *threat*.

The major disadvantage of this approach is that it has a lot of parameters which need to be tuned. It suits well the problems, in which the weight can be naturally interpreted as a similarity between the two connected entities.

2.2 WHOIS analysis

The WHOIS records has been mostly used as a source of additional features to train classifiers, as for example Support Vector Machines or Decision tree algorithm. Although some of the articles showed more creative way to use them. Same as the DNS data, the WHOIS records can be used to find connections between otherwise unrelated domains based on common registrant information. In this section we will review the usage of WHOIS records in the recent articles.

2.2.1 We know it before you do

Xu et al. have described the life-cycle of a malicious domain [19]. According to their study it consists of three main phases:

1. **Preparation** name of the domain is selected, registered and the DNS record is set up
2. **Activation** the period of time in which the domain is actively used
3. **Deactivation** deactivation and parking for future reuse or reselling of the domain

The length of each phase varies, but they showed that before activation a domain might be kept unused for months. It means that there is still a time window in which we can identify a domain which has not been used yet.

The authors also proposed a solution to how to predict future usage of malicious domains. It consists of four different heuristic approaches:

- **Estimation of re-use of malicious domains** Based on a scoring function, which is not described in the text, the authors estimate the possibility of re-using of known malicious domain, which has been already deactivated.
- **Reverse engineering of DGA algorithms** Using a sand-boxing technique the authors to identify malware samples which uses DGA algorithm to generate set of possible rendezvous domains. For the malware samples which do use it, the authors claim to generate a list of domains which will be used in the future.
- **Observed patterns in DNS queries** Based on observed patterns in DNS queries, as for example CNAME record pointed to some known malicious domain, authors identify other possibly malicious domains.
- **Connections between malicious domains** The authors use WHOIS records to identify domains which share registrant information with some known malicious domains. They also use passive DNS data to identify domains which are hosted on a same server as some known malicious domains.

All of the approaches are explained very vaguely, leaving out details important for their understanding. Therefore it is not possible to reproduce, or verify their results.

2.2.2 Predictive domain blacklisting

Felegyhazi et al. analyzed the potential of proactive domain blacklisting [12] They observed that malicious domains are often registered in bulk. Based on this observation they find previously unknown malicious domains which were co-registered with some known malicious domains on the same day, sharing the same nameserver.

They have also mentioned that malicious domains often migrate between name servers. This is not usual for benign domains, their network profile is often settled. If a known malicious domain switches its authoritative nameserver from a to b , they look for another domains which changed from a to b at the same time.

Using this heuristic approach they collect suspicious domains, which are potentially malicious. Then they group those domains based on their WHOIS records. Domains in groups which contain some known malicious domain are assumed to be malicious as well.

This approach takes into account primarily the correlations of registrations of malicious domains in time and the WHOIS records are used only to prevent false positives. The authors claim to have achieved 93% precision, from which 73% are domains which were blocked later after the discovery and 20% are domains flagged suspicious by the McAfee SiteAdvisor. The remaining 7% of domains are not labeled as false positives, but as "unknown".

2.2.3 Meta-data driven classification

One of the possible ways to detect a connection to malicious website is by analysis of the URL from the connection. Most of the previous research in this area focused on creating novel features which would discriminate between malicious and benign domains. Some of the articles also mention using WHOIS records as one of the sources of the features [18, 15].

Ma et al. showed in [15] that registration date of a domain and the time of the last update of its WHOIS record are relevant features. They also used the WHOIS records to extract information about registrar, and registrant and converted it into categorical features. A lot of those featured turned out to be relevant, because some of registrars are know to host malicious content more frequently then others.

Those approaches seems to be working well during experimental phase but are hardly applicable in the real world. It is due to the limited sizes of datasets which are used for the training and evaluation. It is not easily possible to train a classifier which would be able to capture all possible values of registrant names, organization names etc. and keep it up to date. Current state of technology render those techniques to be unusable in practice.

2.3 Reputation system based on observed evidence

Reputation is a term previously used in social and economical studies. In the days of the e-commerce, the reputation based systems were studied as a possible way of ranking buyers and sellers on the Internet.

Jøsang et al. proposed a reputation system called Beta Reputation system [10]. The reputation was represented as an estimated probability that a seller gets a positive feedback. They modeled the posterior probability of the reputation using Beta distribution, which also served as an uniform prior. The reputation is then simply defined as

$$\psi(n, p) = E_x[\text{Beta}(n + 1, p + 1)] = \frac{n + 1}{n + p + 2} \quad (2.3)$$

where the p is the number of positive feedbacks and n is the number of negative feedbacks.

Years later West at al. used this system to calculate reputation of domains on the Internet [18]. The calculated value was then used as one of features to train a classifier of URLs to malicious / legitimate class. The authors showed that the reputation had the highest information gain of all the features which were used.

Subsequently it can be useful to track reputation of domains to get an overview about their maliciousness. Problem of such system is that it can not give an opinion about a domain which was not observed before. We will target this problem by introducing a system which do not only take into account the observed evidence to give the opinion about a domain. We can therefore a priori estimate a maliciousness of a domains without the need to actual observe them.

Chapter 3

Proposed Solution

The goal of this thesis is to create a system which would be able to estimate a probability that a domain d is malicious, i.e. maliciousness of the domain m_d , based on observed evidence and data stored in WHOIS records. As it was shown in the introduction, malicious actors often reuse email addresses and other resources to register multiple domains. If extracted from the WHOIS records, such keys might connect otherwise unrelated domains, which were possibly registered by the same registrant. Those keys must be well discriminative, in our case we use following four types of keys

- k_n registrant's name
- k_o organization
- k_e contact email address
- k_p postal address

For each domain d we therefore have quartet $K_d = (k_{n_d}, k_{o_d}, k_{e_d}, k_{p_d})$. The other fields, as for example authoritative nameserver of a domain, can provide some information about maliciousness of the domains as well, but they can not be used to distinguish different registrants.

The relationships between the domains and the keys can be described using bipartite graph, shown in Figure (3.1). The graph consists of observed domains O , unobserved domain U , and keys K . The difference between the O and U is that we have both evidence and keys for the domains in O , but only keys for the domains in U . Our goal is to create a reputation system, which would be able to give an opinion about a domains from both O and U .

Information in WHOIS records is very coarse-grained because the records are available only for the registered domains, such as `example.com`, but not for it's sub-domains. It therefore can not be directly used to classify domains into malicious / benign class. But it can be used to get a prior estimate about maliciousness of a domains d given it's keys K_d , which is our goal.

To be able to create the desired system we first need to build a ground-truth. The data we have available come from a major Intrusion Prevention System (IPS). It consists of a few terabytes of proxy-logs captured during the first week of several subsequent months. The logs consists of information about time, target IP, client IP, URL, HTTP header and

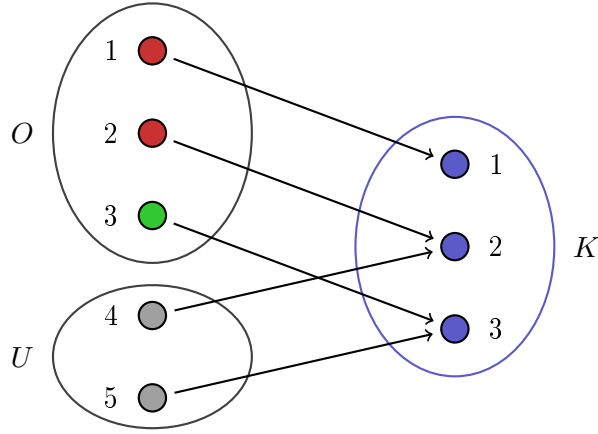


Figure 3.1: Bipartite graph of keys and domains. The domains are on the left side, and the keys are on the right. The domains consists of two sets, the set of observed domains, and the set of unobserved domains. For the observed domains we have both keys and evidence, but for the unobserved keys we have only their keys. The red nodes are domains for which we observed mostly malicious behaviour, the green are domains for which we have observed legitimate behaviour.

several other fields including a flag signifying whether the connection was blocked or not. In each log, we are interested in the domain name d from the URL and the flag b whether the connection was blocked. The data can be therefore represented as tuples $(b, d) \in T$.

The IPS blocks access to malicious domains based on the latest intelligence feeds and near real-time analysis of the traffic. We use this labeling as a ground truth because the system aggregates both publicly available and commercial blacklists plus information from it's own analytic systems.

It is also important that the data capture the state of belief about the maliciousness of domains at the time of the connections. This property is important and we will use it during the evaluation of the proposed algorithm. It allows us to train the system on observed data from one month and then evaluate the performance on actually observed evidence from the following month. Comparing our approach to the usual way of splitting data to training and testing sets, this approach captures the true predictiveness of the evaluated algorithm, making the results more trustworthy.

3.1 Proposed model

Let us assume that the probability that an observed connection is blocked, given the target domain d , follows the Bernoulli distribution which is defined as

$$p(b|d, M) = m_d^b (1 - m_d)^{1-b}, \tag{3.1}$$

where $M = \{m_d\}_{d \in D}$ is the set of probabilities of maliciousness for all domains $d \in D$. The data is assumed to be i.i.d. and so the m_d can be interpreted as the relative portion of

blocked connections targeting the domain d . To learn the m_d we first need to specify the likelihood of observing of the set of connections T which is

$$p(T|M) = \prod_{(b,d) \in T} p(b|d, M) = \prod_{(b,d) \in T} m_d^b (1 - m_d)^{1-b}. \quad (3.2)$$

We can easily find a Maximum Likelihood (ML) estimate of the m_d for each $d \in O$ as

$$m_{d\text{ML}} = \arg \max_{m_d} \log p(T|M) = \frac{N_{\text{blocked}}(d)}{N_{\text{allow}}(d) + N_{\text{block}}(d)}, \quad (3.3)$$

where the $N_{\text{allow}}(d)$, $N_{\text{block}}(d)$ is the number of blocked, allowed flows targeting the domain d respectively. This estimate is not really useful, because it tends to heavily over-fit the data and does not generalize to previously unknown domains, which is our goal. Take for example the case when we observe a single blocked connection to the domain d . We know almost nothing about the domain, but the maliciousness m_d is already estimated to one. This problem has to be dealt with, because as you can see in Figure 3.2, even with the amount of data we have available, we often observe only a very few samples per domain.

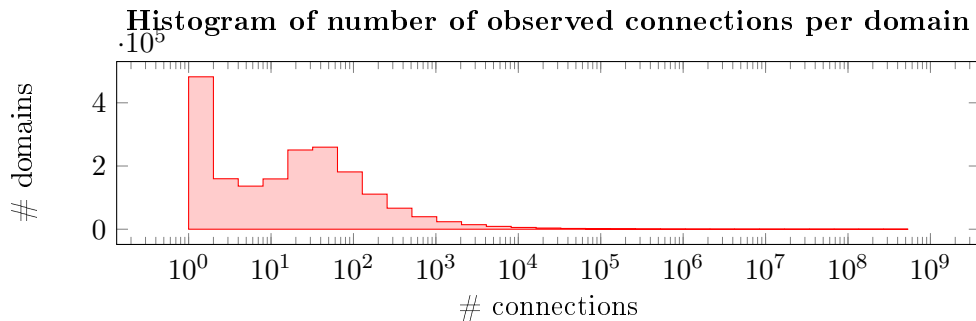


Figure 3.2: Histogram of number of observed connections. The data was collected during the first week of September 2014. You can see that we have observed only single connection to over a half of million unique domains. The most frequently accessed domains had over a 100M of unique connections during the week. On average a domain is accessed 575 times per day.

To avoid this problem we take the Bayesian approach to statistics, which is to treat all sources of uncertainty as random variables [13]. We therefore assume that all the m_d are also random variables, which allows us to introduce a prior distribution to them. We choose a Beta distribution, because it is a conjugate prior to the Binomial distribution. It is defined as

$$\text{Beta}_x(a, b) = \frac{x^{a-1}(1-x)^{b-1}}{\beta(a, b)}, \quad (3.4)$$

where $a, b \in \mathbb{R}^+$ and the

$$\beta(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} \quad (3.5)$$

is a normalization coefficient and the $\Gamma(x)$ is the factorial $x!$ generalized to real numbers.

In general a prior is conjugate to a likelihood if the posterior distribution has the same form as the prior distribution. Conjugacy is an important property as it ensures that the posterior has a known form. This is crucial for further analytical operations and analysis, because the properties and moments of such distribution are known. Also as the form of the prior and the posterior is the same, the posterior can be continually updated by the Bayes rule, as more evidence is observed.

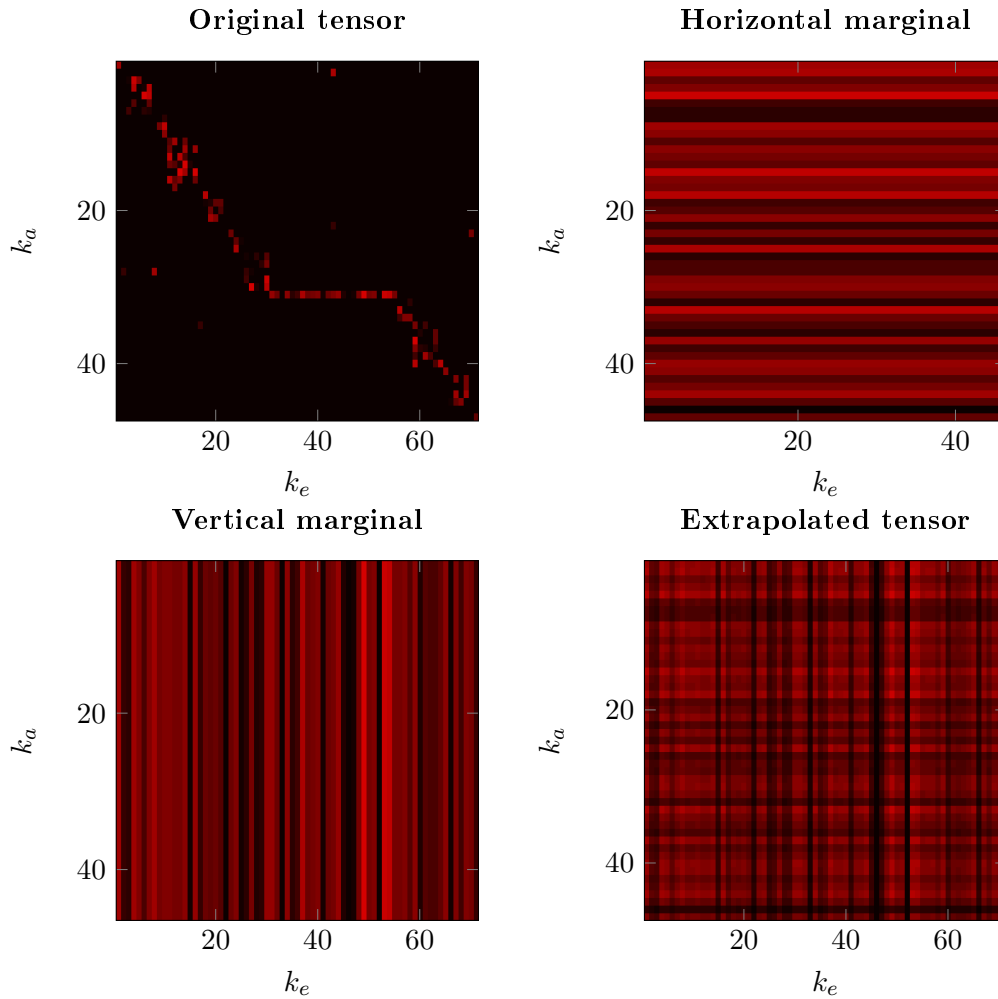


Figure 3.3: Graphical representation of extrapolation of the partially observed tensor. The observed combinations of keys K_d are represented by red color. The brightness of the color is proportional to the value of the tensor. The black areas are the combinations of K_d which have not been observed before. The tensor is extrapolated as a product of two marginals, which best fit the actually observed values.

As we have said before, we would like to have a prior estimate of maliciousness m_d of a domain d based on it's WHOIS record. To do so we let the parameters of the prior a and b to be functions of the set of keys K_d . Because the keys can be divided into groups according

to their type and enumerated we can also think of a and b as of tensors, where the K_d are the coordinates of the appropriate values.

In Figure 3.3 you can see a graphical representation of the tensor a for some of the observed combinations of K_d . To be able to plot it we have selected only two axis out of the four, particularly the k_e and k_a . The black colored areas are the combination of K_d which were not observed in the available data.

Our goal is to extrapolate the information from the observed combinations of keys to all of the possible combinations. The approximation must be simple because we want the final inference algorithm to produce closed form, tractable results. One possible way to do it is by decomposition of the a_{K_d}, b_{K_d} as

$$a_{K_d} = \prod_{k \in K_d} a_k \quad (3.6)$$

$$b_{K_d} = \prod_{k \in K_d} b_k, \quad (3.7)$$

where the values a_{K_d}, b_{K_d} are approximated as products of marginal values a_k, b_k for all $k \in K_d$. The intuition behind the a_k and b_k is the same as behind the composed values, they provide us some prior estimate about maliciousness of the key k .

Let us analyse the original tensor in Figure 3.3. The address $k_a = 31$ is used together with all the emails $k_e \in [31, 55]$ to register some domains. It might belong to a P.O. box of an organization offering an anonymization service. Those services usually assign unique email for each domains so the registrant is able to receive communication, but all the other keys are shared by all of the other registered domains. As you can see in the extrapolated tensor, those values are approximated well, with almost no details blurred.

On the other hand the group of pairs of keys around the $k_e = k_a = 15$ is more tangled up between each other and therefore the marginalization might blur out some details. Those cases are fortunately rare. Also we might ask how likely is it that those domains were not registered by the same registrant. Hence the blur is not necessarily a bad thing, because it propagates information across the keys. Moreover it only influences the prior estimates, which should fade away as the actual evidence is obtained.

Since we have already defined the prior distribution to m_d we can proceed in composing the model. The posterior distribution is proportional to the product of the prior and likelihood as

$$p(M|A, B, T) \propto p(T|M) p(M|A, B), \quad (3.8)$$

where the A, B are the sets of a_k, b_k for all $k \in K$ respectively and the $p(M|A, B)$ is a product of prior distributions of m_d for all $d \in D$ defined as

$$p(M|A, B) = \prod_{d \in D} p(m_d|A, B) \quad (3.9)$$

$$= \prod_{d \in D} m_d^{a_{K_d}-1} (1 - m_d)^{b_{K_d}-1}. \quad (3.10)$$

By substituting (3.10) to (3.8) we get

$$p(M|A, B, T) \propto p(T|M) p(M|A, B) \quad (3.11)$$

$$= \prod_{(b,d) \in T} m_d^b (1 - m_d)^{1-b} \prod_{d \in D} m_d^{a_{K_d}-1} (1 - m_d)^{b_{K_d}-1} \quad (3.12)$$

$$= \prod_{d \in D} m_d^{N_{block}(d)+a_{K_d}-1} (1 - m_d)^{N_{allow}(d)+b_{K_d}-1} \quad (3.13)$$

$$\sim \text{Beta}(N_{block}(d) + a_{K_d}, N_{allow}(d) + b_{K_d}) . \quad (3.14)$$

Unfortunately, because of the normalizing coefficient of the Beta distribution (3.5), the ML estimate of the parameters a_k, b_k is not tractable. Instead, following again the Bayesian approach, we choose to model the parameters as random variables and introduce prior distributions to them.

In the (3.14) you can see that the values directly compete with the observed evidence in a sense, that the greater the value of the a_{K_d}, b_{K_d} is the more observed evidence we need to make the prior insignificant. Therefore the priors, which we are about to introduce, should restrict the a_k, b_k to have some "reasonable" value. For example, we might want take into account the evidence only if the number of observed connections targeting a domain is larger than 10. On the other hand the priors should not interfere with the inference process. By setting too strong priors the inference might be forced to learn values close to the specified mean, ignoring the observed evidence. The strength of a prior is determined by its variance, which should be therefore large. The exact choice of the parameters will be further discussed in Section 4.5.

Unfortunately the Beta distribution does not have any standard conjugate prior. Ma et al. proposed an *Extended Factorisation Approximation* based on a Variational Bayes method, to solve this problem while estimating parameters of Beta Mixture Models [16] . By following their approach we can we assume that for all $k \in K$ the a_k, b_k are independent, mathematically speaking

$$p(A, B) \approx p(A) p(B) = \prod_{k \in K} p(a_k) p(b_k) . \quad (3.15)$$

Then any well known distribution defined on interval $[0, \infty)$ can be used as the prior. For the Variational Bayes method to be feasible we have to ensure that the posterior distribution is conjugate to the factorized prior. If it was not then we would have to deal with possibly unknown forms of distributions, which would make analytical operations with the distribution difficult. Ma showed that if we choose a Gamma distribution as the prior

$$p(a_k) \sim \text{Gamma}(u_a, v_a) = \prod_{k \in K} \frac{v_a^{u_a}}{\Gamma(u_a)} a_k^{u_a-1} e^{-v_a a_k} \quad (3.16)$$

$$p(b_k) \sim \text{Gamma}(u_b, v_b) = \prod_{k \in K} \frac{v_b^{u_b}}{\Gamma(u_b)} b_k^{u_b-1} e^{-v_b b_k} , \quad (3.17)$$

then under certain approximations, as it will be described later in Section 3.2.2, the approximated posterior distributions $q(a_k), q(b_k)$ are also Gamma.

We do not need to a priori distinguish between the different keys, thus the parameters of the Gamma prior are set to be the same for all $k \in K$. The values of the parameter $\Theta = (u_a, v_a, u_b, v_b)$ are therefore the only four parameters of the model which need to be tuned.

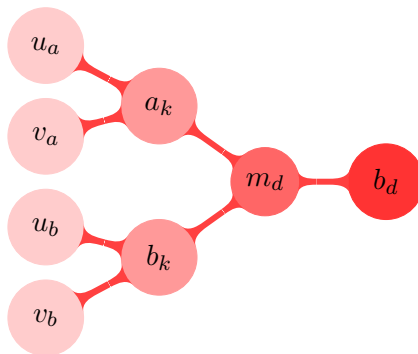


Figure 3.4: Dependency of the proposed variables described using a graph. Each two adjacent variables are dependent. Variable on the right side always depends on the variable on the left side of the edge, which is also represented by the fading colour.

In Figure 3.4 you can see the dependencies between the proposed variables. The dependency is represented by the fading color and it goes from right to left. We observe the tuples $(b, d) \in T$, or b_d . A block b_d given the domain d depends on the maliciousness of the domain m_d . The m_d depend on the values a_k, b_k of the keys of the domain K_d . And finally the a_k depend on the parameters of the Gamma prior u_a, v_a and similarly for the b_k .

3.2 Inference using Variational Bayes

The complete model is then defined as

$$p(M, A, B|T) \propto p(M, A, B, T) \quad (3.18)$$

$$= p(T|M) p(M|A, B) p(A) p(B), \quad (3.19)$$

which is also not tractable, because the integral

$$c(T) = \int_{M \in \mathcal{M}} \int_{A \text{ in } \mathcal{A}} \int_{B \in \mathcal{B}} p(M, A, B, T) dB dAdM, \quad (3.20)$$

does not have an analytical solution.

In theory the posterior distribution could be evaluated numerically, as for example by Gibbs sampling. Unfortunately those methods are computationally demanding making them impractical for large datasets as we have. We therefore focus on approximating it using the VB method by finding marginals $q(m_d), q(a_k), q(b_k)$ such as

$$p(M, A, B|T) \approx q(M, A, B) = \prod_{d \in D} q(m_d) \prod_{k \in K} q(a_k) q(b_k). \quad (3.21)$$

As described in Appendix A, the VB is an iterative method, which minimizes the Kullback–Leibler (KL) divergence between original distribution $p(M, A, B|T)$ and the factorized distribution $q(M, A, B)$ by setting the approximate marginals to

$$\log q(b_k) \propto E_{M,A,B \setminus b_k}[\log p(M, A, B|T)] \quad (3.22)$$

for all $k \in K$, and similarly for the $q(m_d)$ and $q(a_k)$.

Now we want to derive the marginal distributions. The very first step is to write down the logarithm from the right hand side of the (3.22), which is

$$\log p(M, A, B|T) \propto \sum_{(b,d) \in T} b \log(m_d) + (1-b) \log(1-m_d) \quad (3.23)$$

$$+ \sum_{d \in D} (a_{K_d} - 1) \log(m_d) + (b_{K_d} - 1) \log(1-m_d) - \log \beta(a_{K_d}, b_{K_d}) \quad (3.24)$$

$$+ \sum_{k \in K} (u_a - 1) \log(a_k) - v_a a_k + (u_b - 1) \log(b_k) - v_b b_k. \quad (3.25)$$

The whole probability distribution is composed of pairs of distributions from the exponential family. It holds that a logarithm of a distribution from exponential family is a sum of products of functions, each of which is a function of only a single variable. This is crucial for the VB method because we can simply replace occurrences of the functions in the products by their expected values, as long as the variables are independent. Without loss of generality we can demonstrate it on the case of two independent variables X, Y and functions $g(x), h(y)$ as

$$E_{x,y}[g(x) h(y)] = E_x[g(x)] E_y[h(y)] = \widehat{g(x)} \widehat{h(y)}. \quad (3.26)$$

3.2.1 Marginals of the m_d

To get the marginal of m_d we have to apply the expectation operator from (3.22) to integrate out all the variables except of m_d , which leads to

$$\log q(m_d) \propto \sum_{(b,d) \in T} b \log(m_d) + (1-b) \log(1-m_d) \quad (3.27)$$

$$+ \sum_{d \in D} (\widehat{a_{K_d}} - 1) \log(m_d) + (\widehat{b_{K_d}} - 1) \log(1-m_d) - \widehat{\log \beta(a_{K_d}, b_{K_d})} \quad (3.28)$$

$$+ \sum_{k \in K} (u_a - 1) \widehat{\log(a_k)} - v_a \widehat{a_k} + (u_b - 1) \widehat{\log(b_k)} - v_b \widehat{b_k}. \quad (3.29)$$

As it was described in Section 1.1.1 the summands which are constants are simply part of the normalizing coefficient of the distribution. Therefore the terms which now became constants w.r.t. the m_d can be omitted.

After reordering the rest of the terms we get

$$\log q(m_d) \propto \log(m_d) \left(\widehat{a_{K_d}} + \sum_{b:(b,d) \in T} b \right) + \log(1-m_d) \left(\widehat{b_{K_d}} + \sum_{b:(b,d) \in T} (1-b) \right), \quad (3.30)$$

where the $\widehat{a_{K_d}}, \widehat{b_{K_d}}$ are products of the expected values of a_k, b_k as

$$\widehat{a_{K_d}} = \prod_{k \in K_d} \widehat{a_k}, \quad \widehat{b_{K_d}} = \prod_{k \in K_d} \widehat{b_k}. \quad (3.31)$$

This equation might seem familiar to you, because it is a logarithm of a Beta distribution. So the marginal $q(m_d)$ can be identified to be

$$q(m_d) \sim \text{Beta}(a_d, b_d), \quad (3.32)$$

where the hyper parameters a_s, b_s are

$$a_d = \widehat{a_{K_d}} + \sum_{b:(b,d) \in T} b, \quad b_d = \widehat{b_{K_d}} + \sum_{b:(b,d) \in T} 1 - b. \quad (3.33)$$

Interestingly, you can see that the form of the approximated marginal is the same as the exact posterior in the model with no Gamma prior applied (3.14). Even though we assumed the m_d to be independent of a_k , and b_k the marginals approximated by VB take into account both prior and observed evidence as we would expected.

3.2.2 Marginals of the a_k

Derivation of this marginal is more complicated. By applying of the expectation operator from (3.22) we get

$$\log q(a_k) \propto \sum_{(b,d) \in T} b \widehat{\log(m_d)} + (1 - b) \widehat{\log(1 - m_d)} \quad (3.34)$$

$$+ \sum_{d \in D} (a_k \widehat{a_{K_d \setminus k}} - 1) \widehat{\log(m_d)} + (\widehat{b_{K_d}} - 1) \widehat{\log(1 - m_d)} - \mathbb{E}_{B,A \setminus a_k} [\log \beta(a_k a_{K_d \setminus k}, b_{K_d})] \quad (3.35)$$

$$+ \sum_{k \in K} (u_a - 1) \log(a_k) - v_a a_k + (u_b - 1) \log(\widehat{b_k}) - v_b \widehat{b_k}, \quad (3.36)$$

where the $\widehat{a_{K_d \setminus k}}, \widehat{b_{K_d \setminus k}}$ are products of the expected values of a_k, b_k as

$$\widehat{a_{K_d \setminus k}} = \prod_{k' \in K_d \setminus k} \widehat{a_{k'}}, \quad \widehat{b_{K_d \setminus k}} = \prod_{k' \in K_d \setminus k} \widehat{b_{k'}}. \quad (3.37)$$

After omitting constants and reordering of what is left we obtain

$$\log q(a_k) \propto \sum_{s:k \in K_d} a_k \widehat{a_{K_d \setminus k}} \widehat{\log(m_d)} - \mathbb{E}_{B,A \setminus a_k} [\log \beta(a_k a_{K_d \setminus k}, b_{K_d})] \quad (3.38)$$

$$+ (u_a - 1) \log(a_k) - v_a a_k. \quad (3.39)$$

You can see that we could not replace the coefficient $\log \beta(a_k a_{K_d \setminus k}, b_{K_d})$ by it's expected value, because the a_k could not be separated from the rest of the a_{K_d} and b_{K_d} .

Ma showed that if we find an unnormalized lower-bound to the $\log p(M, A, B|T)$ such that

$$\log p(M, A, B|T) \geq \log p_{LB}(M, A, B|T), \quad (3.40)$$

then the lower-bound $\mathcal{L}(q)$ from (A.3) is also lower-bounded as

$$\mathcal{L}(q) = \int q(a_k) \log \frac{p(M, A, B|T)}{q(a_k)} da_k \quad (3.41)$$

$$\geq \int q(a_k) \log \frac{p_{LB}(M, A, B|T)}{q(a_k)} da_k, \quad (3.42)$$

and therefore by substituting the $p_{LB}(M, A, B|T)$ into (3.22)

$$\log q(a_k) \propto \mathbb{E}_{M, B, A \setminus a_k} [\log p_{LB}(M, A, B|T)], \quad (3.43)$$

we can maximize the lower-bound of the $\mathcal{L}(q)$. Ma stated that although we can not maximize the $\mathcal{L}(q)$ directly, by maximizing its lower-bound we can reach an optimum of the $\mathcal{L}(q)$ asymptotically.

The only term in (3.38) which is not tractable is $-\mathbb{E}_{B, A \setminus a_k} [\log \beta(a_k a_{K_d \setminus k}, b_{K_d})]$, so we need to find the lower-bound for it. Also we want the approximate posterior to have a form of a Gamma distribution, so it is conjugate to the prior. If the posterior has a standard form, the expected values of a_k have a known, closed form solutions making the iterations of VB easy to proceed.

The expression (3.39) is the logarithm of the Gamma prior to a_k . You can see that the only way for the lower-bounded term to become a part of a hyper-parameter of the posterior distribution is to have a form of $\zeta \log a_k$, where the ζ is a scalar constant w.r.t. the a_k . It is because the ζ would then get summed up with the u_a to create the posterior hyper-parameter u_{a_k} . As we will show in the Section 3.2.4 the term can be lower-bounded as

$$-\mathbb{E}_{B, A \setminus a_k} [\log \beta(a_k a_{K_d \setminus k}, b_{K_d})] \geq \zeta_{d, a_k} \log(a_k), \quad (3.44)$$

where $\zeta_{d, k}$ is a constant which depends on the key k of and a domain d .

By substituting (3.44) into (3.39) we can identify the final form of the posterior as

$$q(a_k) \sim \text{Gamma}(u_{a_k}, v_{a_k}), \quad (3.45)$$

where the hyper-parameters u_{a_k} and v_{a_k} are

$$u_{a_k} = u_a + \sum_{s: k \in K_d} \zeta_{d, k} \quad (3.46)$$

$$v_{a_k} = v_a - \sum_{s: k \in K_d} \widehat{a_{K_d \setminus k}} \widehat{\log m_d}. \quad (3.47)$$

3.2.3 Marginals of the b_k

The derivation of the marginal $q(b_k)$ is almost identical to the $q(a_k)$ so we will only present you the result, which is

$$q(b_k) \sim \text{Gamma}(u_{b_k}, v_{b_k}), \quad (3.48)$$

where the hyper-parameters u_{b_k} and v_{b_k} are

$$u_{b_k} = u_b + \sum_{s:k \in K_d} \zeta_{d,k} \quad (3.49)$$

$$v_{b_k} = v_b - \sum_{s:k \in K_d} \widehat{b_{K_d \setminus k}} \widehat{\log(1 - m_d)}. \quad (3.50)$$

3.2.4 Approximation of the normalizing coefficient

We need to find a lower-bound for the function $Q(a_k) = -\mathbb{E}_{B,A \setminus a_k}[\beta(a_k, a_{K_d \setminus k}, b_{K_d})]$. Ma has already provided a lower-bound for $Q'(a) = -\mathbb{E}_b[\beta(a, b)]$, which is insufficiently general for us to use directly, because the a, b are only scalar variables. In our case the parameters are not scalars, but products of multiple variables. By following the derivation steps from the original paper we will find the lower-bound generalized for $Q(a_k)$.

The following two properties are necessary to derive the lower-bound.

Property 1. *The normalization coefficient of Beta distribution can be approximated in point x_0 using pseudo-Taylor approximation as*

$$-\log \beta(x, y) \geq -\log \beta(x_0, y) + [\psi(x_0 + y) - \psi(x_0)] x_0 (\log x - \log x_0), \quad (3.51)$$

where the ψ is Digamma function, which is defined as first derivative of a logarithm of Gamma function. This inequality holds for $y > 1$, and $x, x_0 \in \mathbb{R}$.

Property 2. *Digamma function can be approximated in point x_0 using pseudo-Taylor approximation as*

$$\psi(x + y) \geq \psi(x_0 + y) + \psi'(x_0 + y) x_0 (\log x - \log x_0), \quad (3.52)$$

where the ψ' is Trigamma function, which is defined as second derivative of a logarithm of Gamma function. This inequality holds for $y > 1$, and $x, x_0 \in \mathbb{R}$.

The derivation proceeds as follows

$$\mathbb{E}_{B,A \setminus a_k} [-\log \beta(a_{K_d}, b_{K_d})] \quad (3.53)$$

$$\geq \mathbb{E}_{B,A \setminus a_k} [-\log \beta(a_0, b_{K_d}) + (\psi(a_0 + b_{K_d}) - \psi(a_0)) a_0 (\log a_{K_d} - \log a_0)] \quad (3.54)$$

$$\propto \mathbb{E}_{B,A \setminus a_k} [(\psi(a_0 + b_{K_d}) - \psi(a_0)) (\log a_k + \log a_{K_d \setminus k} - \log a_0)] a_0 \quad (3.55)$$

$$\propto \log a_k (\mathbb{E}_B [\psi(a_0 + b_{K_d})] - \psi(a_0)) a_0. \quad (3.56)$$

In the (3.54) we apply the Property 1 to approximate the Beta function in point a_0 . You can see that several terms then became constants w.r.t. the a_k so we omit them in (3.55). Also the term $\log a_{K_d}$ can be rewritten as $\log a_k + \log a_{K_d \setminus k}$, so we separate the a_k from the rest of the variables. Finally in the (3.56) we drop all the remaining constants and move the expectation operator inside the brackets.

To expand the term $\mathbb{E}_B [\psi(a_0 + b_{K_d})]$ we use the Property 2.

$$\mathbb{E}_B [\psi(a_0 + b_{K_d})] \geq \mathbb{E}_B [\psi(a_0 + b_0) + \psi'(a_0 + b_0) b_0 (\log b_{K_d} - \log b_0)] \quad (3.57)$$

$$= \psi(a_0 + b_0) + \psi'(a_0 + b_0) b_0 (\mathbb{E}_B [\log b_{K_d}] - \log b_0), \quad (3.58)$$

where the approximation is done in point b_0 and the expectation operator is moved deeper inside the brackets.

By substituting (3.58) into (3.56) we obtain the desired solution as

$$\mathbb{E}_{B,A \setminus a_k} [-\log \beta(a_{K_d}, b_{K_d})] \quad (3.59)$$

$$\geq \log a_k \{ \mathbb{E}_B [\psi(a_0 + b_{K_d})] - \psi(a_0) \} a_0 \quad (3.60)$$

$$\geq \log a_k \{ \psi(a_0 + b_0) - \psi(a_0) + b_0 \psi'(a_0 + b_0) (\mathbb{E}_B[\log b_{K_d}] - \log b_0) \} a_0 \quad (3.61)$$

$$= \log a_k \zeta_{d,a_k}. \quad (3.62)$$

Points a_0 and b_0 , in which we do the Taylor expansion, must be selected properly. The derived approximation is valid for $a_0, b_0 > 1$. We therefore set the values to be the expected values of a_{K_d}, b_{K_d} from the previous iteration if the values are greater than one, and one otherwise.

3.3 Algorithm in steps

Because of the closed forms of the marginals, the algorithm itself is very simple. As you can see in Algorithm 1, all you need to do is to iteratively recompute the marginals based on the definitions from the previous sections.

Algorithm 1 Variational Bayes inference of parameters of the model

1. choose the parameters $\Theta = (u_a, v_a, u_b, v_b)$
 2. $\forall k \in K$ choose initial estimates of $q(a_k)$ and $q(b_k)$
 3. **For** it = 1 to max_iterations
 - (a) $\forall d \in D$ recompute $q(m_d)$
 $\forall d \in D$ evaluate $\widehat{\log m_d}, \widehat{\log(1 - m_d)}$
 - (b) $\forall a_k \in A$ recompute $q(a_k)$
 $\forall a_k \in A, d \in D$ evaluate $\widehat{a_k}, \widehat{\log a_{K_d}}$
 - (c) $\forall b_k \in B$ recompute $q(b_k)$
 $\forall b_k \in B, d \in D$ evaluate $\widehat{b_k}, \widehat{\log b_{K_d}}$
 - (d) **If** convergence_criteria < threshold **Then** break
-

In the further sections we will discuss the initialization strategy and the choice of the convergence criteria.

3.3.1 Initialization strategy

During the initialization of the inference process an initial estimates of $q(a_k)$ and $q(b_k)$ are needed for all $k \in K$. We will discuss several strategies which might be used.

A seemingly obvious choice is to use a random initialization. This is usual in some other variational algorithms, as for example Expectation Maximization (EM) or k -means. Such strategy might feed misleading input to the algorithm, because the values of a_k and b_k are not independent of each other. In the terms of k -means, this is similar to randomly assigning points to the centroids, instead of random initialization of centroids, which is quite different.

Another possibility is to use the data to create an informative prior. Let us define the malicious ratio of a key r_k as the relative number of blocked connections targeting domains featuring the key k . We might think of the expected values $\widehat{a}_k, \widehat{b}_k$ to be parameters of an imaginary Beta distribution over m_k , which is a maliciousness of the key k . Then from the definition of the expected value of the Beta distribution we can find the expected values of $q(a_k), q(b_k)$ for which hold that

$$\widehat{m}_k = \frac{\widehat{a}_k}{\widehat{a}_k + \widehat{b}_k} = r_d. \quad (3.63)$$

Although informative, this initialization can not provide any additional information to the algorithm, because it is based on the same data which is used during the learning phase.

Finally we can simply use the parameter Θ to initialize the Gamma priors as

$$q(a_k) \sim \text{Gamma}(u_a, v_a) \quad (3.64)$$

$$q(b_k) \sim \text{Gamma}(u_b, v_b) \quad (3.65)$$

for all $k \in K$. This way we let the algorithm to learn purely from the data, not forcing it any additional belief about the values of a_k, b_k other than Θ .

We tried the initialization strategies mentioned above but none of them seemed to have any measurable deterministic impact on the quality of the inferred values. We therefore use the simplest non-informative strategy.

3.3.2 Convergence criteria

As shown in Appendix A the VB method is guaranteed to converge. Therefore any simple stopping condition should be sufficient for the algorithm to determine whether it has already converged or not.

Unfortunately in our case we have encountered some problems with the convergence. In some rare cases it happened that the mean values of a_k converged, but the variance kept decreasing. This problem might be caused by the Taylor approximation, which is used in (3.39). Because we are primarily interested in the expected values of the marginal distributions we can avoid this problem simply by checking whether the maximum change of the expected values between iterations is lower than some threshold.

3.4 Implementation

During the implementation we had to face several challenges, such as memory efficiency and data representation. In this section we will briefly sketch up some of the important solutions, which we have introduced.

3.4.1 Processing WHOIS records

The amount of the available WHOIS records is massive and it therefore needs to be stored in the Hadoop Distributed File System (HDFS) and accessed using Map-Reduce. The processing is done by a cascade of Map-Reduce jobs. The first job finds the sets of keys K_d for all observed domains $d \in O$. This is followed by second job, which for all the keys K_d of domains $d \in O$ finds other domains $u \in U$ such that $K_u \cap K_d \neq \emptyset$.

Unfortunately it often happens that a key connects millions of otherwise unrelated domains. Examples of such keys might be organization names and addresses of P.O. Boxes of anonymization services. The point of the keys is to connect together domains which might be registered by the same registrant to get some prior estimate about their maliciousness. Such keys clearly do not satisfy this property and we can therefore prune them. Without the pruning the amount of domains in U would be so high that we would not be able to process it further.

The output of the second job has a format

$$k \rightarrow (O_k, U_k), \quad (3.66)$$

where the O_k , and U_k are the observed, and other domains which feature the key k respectively.

3.4.2 Compact graph structure

Any graph algorithm which runs in at least $\Omega(|E|)$ might not be feasible on the complete bipartite graph of domains D and keys K . To reduce the number of edges and vertices we have created a compact representation of the graph. It groups together all domains sharing the same combination of keys κ .

The first step is to invert the relation (3.66) as

$$d \rightarrow (o, K_d), \quad (3.67)$$

which is a mapping from domains to a tuple, where the o is the flag specifying, whether the domain belongs to O or U , and the keys of the domain K_d . This representation resembles the original WHOIS records but it is not compact at all. The trick is in inverting this relation once again such that we get

$$\kappa \rightarrow (O_\kappa, U_\kappa), \quad (3.68)$$

which is a mapping from all unique combinations of keys κ to the observed, other domains which were registered using them.

The idea behind this representation is that in the case of the unobserved domains U_κ all the $u \in U_\kappa$ are indistinguishable. It is because all we know about them are the keys which they share. The domains therefore should share the same inferred label. Hence we can save time by focusing on inferring a single label for each κ instead of for all domains d .

3.4.3 Parallelization

Algorithms defined on the graph composed of domains and keys can be easily parallelized. Using the compact graph representations described in the previous section, we can find all connected components of the graph. The connected components are independent of each other, because any two domains from different components can not share any key and therefore can not be related. Thus the inference algorithms can run separately on each of the components.

3.4.4 Problems with the approximation

Because of the Taylor approximation, it sometimes happens that the hyper-parameter

$$u_{a_k} = u_a + \sum_{s:k \in K_d} \zeta_{d,a_k} \quad (3.69)$$

of the Gamma prior from (3.45) is negative. We deal with this problem simply by setting the value u_{a_k} zero if the result was lower than zero and keeping it otherwise.

3.4.5 Handling missing keys

It is not unusual that a key is missing. Actually as you can see in Table 3.1 approximately half of the domains have at least one key missing. This is due to the fact that we process a large amount of data and therefore we have to prune frequent keys.

Number of keys	1	2	3	4
Number of domains	77932	44838	135836	221705

Table 3.1: Number of keys per domain

A solution might be to create a single key k'_t for each type t and substituting it as a replacement for all the missing keys of the type. This would have undesirable effects, because it would connect a lot of the unrelated keys. Also the values of $a_{k'_t}$, $b_{k'_t}$ would contribute to the composed values a_{K_d} , b_{K_d} and the values would get blurred.

To avoid this problem we might introduce a substituting keys $k'_{t,d}$ unique to all the domains. This would solve the blurring problem, because the substituted keys would contribute only to values of its own domain d . On the other hand it would cause massive increase in the overall number of keys, which would slow down the already computationally intensive operations with the data. Also the compact graph representation from the Section 3.4.2 would not be compact anymore, because the unique sets of keys κ would be unique to each of the domains.

This leads us to a compromise, which we use in our experiments. Each node of the compact graph consists of a unique combination of keys κ and the domains which share it D_κ . Then if there are some missing keys in the κ , the supplied keys $k'_{t,d}$ should be shared by all domains in D_κ , but no other. This way only the most related domains would be connected by the newly introduced keys. You can see an example of this approach in Figure 3.5.

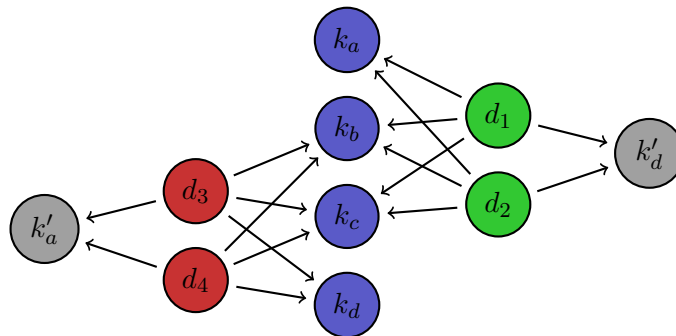


Figure 3.5: Supplying missing keys. You can see that the substitutional keys k' are supplied to groups of domains which share the same set of keys κ .

Chapter 4

Evaluation

4.1 Empirical analysis of convergence

The basic properties of the algorithm can be shown on a simple example with a single domain d and a key k . We initialize it with 50 observed connections from which 80% was blocked. The complete list of parameters can be found in Table 4.1.

Parameter	u_{a_0}	v_{a_0}	u_{b_0}	v_{b_0}	N_{blocked}	N_{allowed}
Value	1	0.2	2	0.2	40	10

Table 4.1: Initial parameters of the algorithm

The a_k and b_k were initialized directly from the definition of mean value of Gamma distribution as

$$a_{k_0} = \frac{u_{a_0}}{v_{a_0}} = 5, \quad b_{k_0} = \frac{u_{b_0}}{v_{b_0}} = 10, \quad (4.1)$$

and the m_d from the definition of mean value of the Beta distribution as

$$m_{d_0} = \frac{a_{k_0}}{a_{k_0} + b_{k_0}} = \frac{1}{3}. \quad (4.2)$$

Then the iterations proceeds as described in Algorithm 1 and the convergence is shown in Figure 4.1. You can see that in this simple setup the algorithm converges rather quickly. The m_d -prior curve is the inferred prior probability that the domain d is malicious, which does not take into account the observed evidence. The m_d -posterior is the inferred posterior probability. Both of the probabilities converged to values very close the malicious ratio r_d , which is defined as the relative number of blocked connections targeting domain d

$$r_d = \frac{N_{\text{blocked}}}{N_{\text{allowed}} + N_{\text{blocked}}}. \quad (4.3)$$

It is also interesting to see to see how the inferred values of m_d depend on the overall number of observed connections N_d if we keep the malicious ratio r_d fixed at 0.8. As you can see in Figure 4.2, with the N_d growing to infinity, the expected value of m_d converge to value close to the malicious ratio r_d . Also the variance of m_d converges to non zero values.

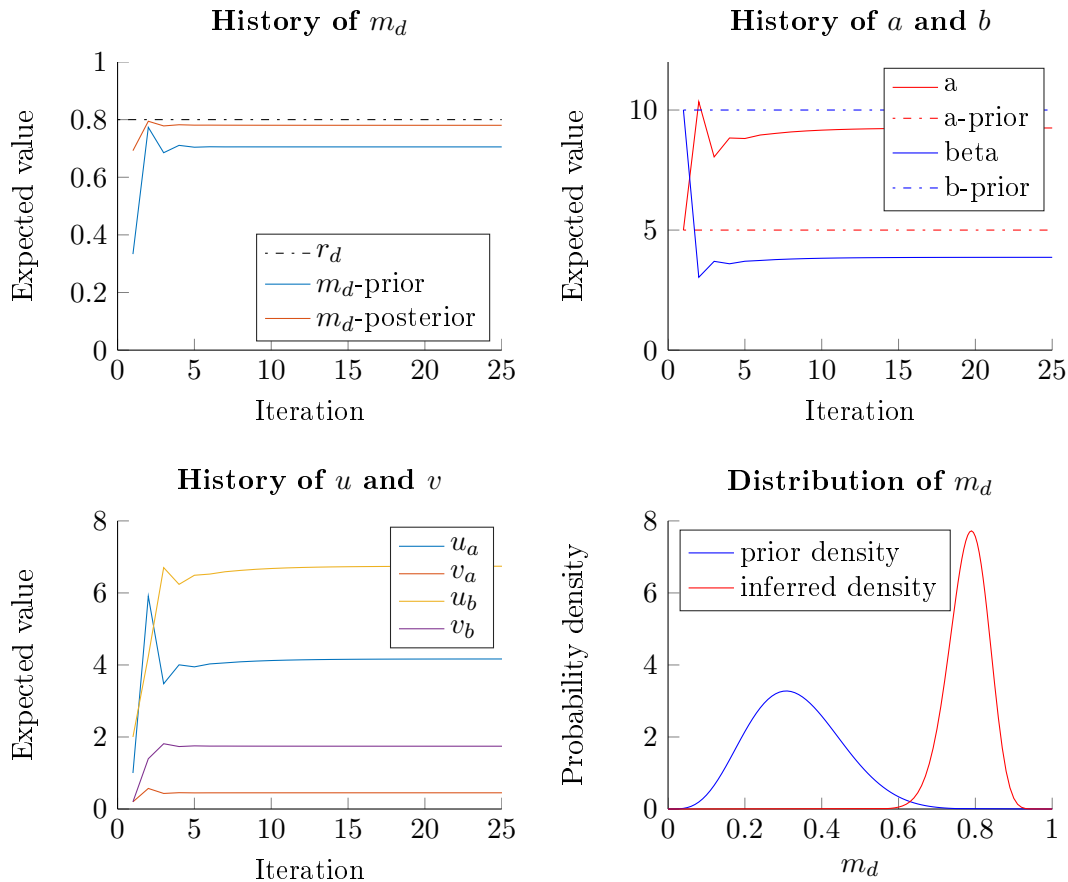


Figure 4.1: Example of convergence of the proposed algorithm. The experimental data consisted of a single key and a single domain. The number of observed connections targeting the domain was 50 from which 40 was labeled as blocked.

This is important, because it signifies that the model does not tend to over-fit the available data. The figure also shows how the variance of the Gamma prior affect the inferred values. You can see that the lower the variance is the stronger the prior is, i.e. the more are the inferred values restricted to the expected values of the prior.

4.2 Datasets

As a part of it's portfolio of security products, Cisco systems offers a Cloud Web Security (CWS) service to enterprise customers. This service provides the customers get near-real-time web protection against potential infections and attacks. If a user inside a covered network tries to access a server which might potentially contain malicious content, the service blocks the request.

As a part of a research team developing anomaly detection system we have access to captures of logs from the CWS. The logs consists of information about time, target IP,

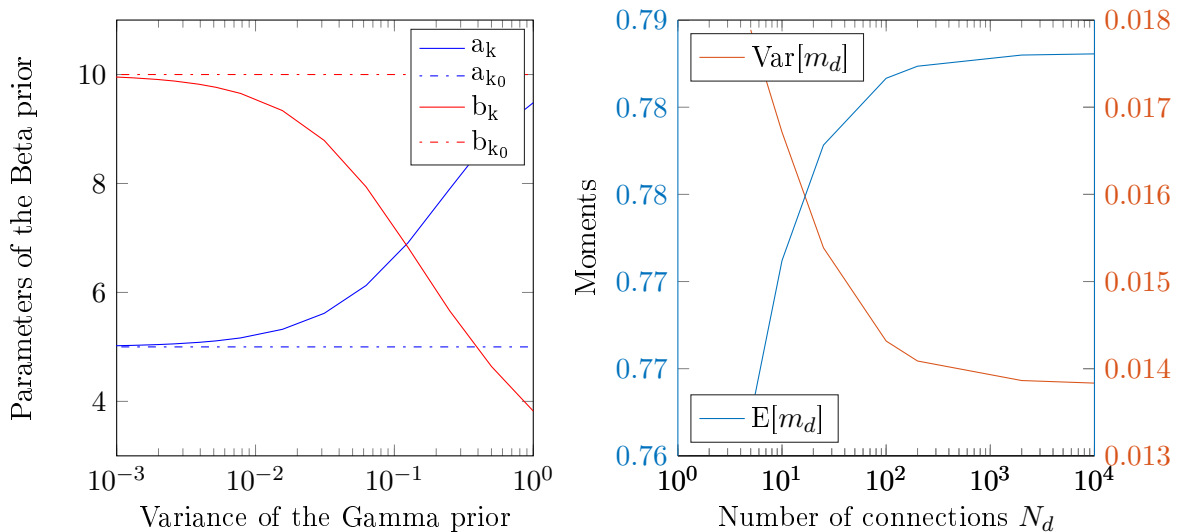


Figure 4.2: An example of convergence of the beta prior. On the left you can see how the actually inferred values of a_k , b_k depend on the variance of the Gamma prior. On the right you can see how the inferred value of m_d converges to the malicious ratio r_d with the growing number of observed flows.

client IP, URL, HTTP header and several other fields including a flag signifying whether the connection was blocked or not. The available data sets consists of captures from the first week of several subsequent months. The quality of the labeling is better than we could achieve by manually collecting publicly available data, because the system utilizes latest commercial and public intelligence feeds. In Table 4.2 you can see the statistics of the available data.

Dataset	# connections	# domains	# domains with blocks
2014-September	7.68G	1.90M	21.9k
2014-October	6.85G	1.78M	12.0k
2014-November	6.71G	1.75M	14.5k
2015-January	7.98G	1.95M	15.9k
2015-February	8.06G	1.97M	16.7k

Table 4.2: Statistics of the available datasets. Intersections of domains between two subsequent months have all approximately 1M domains. Also you can see that approximately 1% of observed domains was connected with some malicious activity.

The form of the data allows us to naturally create training and testing sets. As you can see in Figure 4.3 we can take the observed domains O_{PRED} from the first month. Then we can use the WHOIS records to find all domains U_{PRED} , which share some key with some of the domains in O_{PRED} . If we then infer the probability of maliciousness m_d for all $d \in U_{\text{PRED}}$ we can test the results using the actually observed data from the following month O_{SUCC} . This way the evaluated performance reflects the actual state of belief about domains a month after the training data were captured. In comparison to the usual way of

splitting the available data to training and testing sets, this approach can reflect the true predictiveness of the system.

Please note that to create the training and testing sets we use only the pairs of datasets which come from subsequent months. Also *September dataset* we mean that the September dataset was used to train the algorithm, but *October* dataset was used to test the predictions, etc.

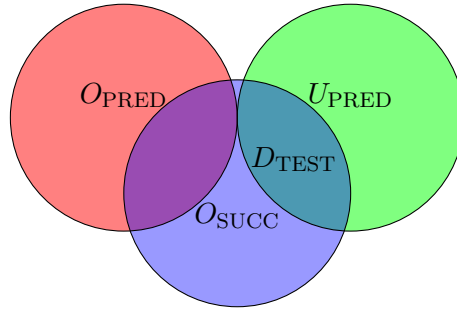


Figure 4.3: Training and testing data explained using Venn Diagram. As it was defined in the beginning of Chapter 3, the O_{PRED} , O_{SUCC} are the domains which we observed in two subsequent time windows, in our case months. The U_{PRED} are the domains which we did not directly observe in the first month, but which share some key with some of the domain from O_{PRED} . We can use the intersection $D_{\text{TEST}} = U_{\text{PRED}} \cap O_{\text{SUCC}}$ to test our predictions from the first month on actually observed evidence in the second month.

The WHOIS records which we have available cover the following top level domains (TLD):

- com, info, net, org, biz, us, asia, mobi, pro, coop

The number of covered TLDs might seem to be insufficient, but if you look at Table 4.3 you will see the relative occurrences of TLDs in the Sophos blacklist. The Sophos is a anti-virus company which daily produces a list of newly emerged threats on the Internet. If you count the relative occurrences, the available WHOIS data cover up to 70% of all known malicious domains, which is more than sufficient for our experiments.

TLD	com	org	ru	net	info	cc	biz	de
Frequency	40.11%	7.68%	7.54%	7.31%	6.69%	6.66%	4.72%	1.54%

Table 4.3: Relative occurrences of TLDs in the Sophos blacklist

4.3 Measures used to evaluate the performance

There are two major points of view from which we need to analyse the proposed algorithm. Firstly we need to discuss the overall precision of the inferred values m_d . Secondly we need to focus on how well does the technique predict the prior on previously unseen domains. To evaluate quality of the inferred prior we use the intersection of domains D_{TEST} as described in the previous chapter.

Using the data from the first month we infer the probabilities m_d . Then simply by selection of a threshold τ we use the inferred probability to classify the observed connections T_{TEST} , which are defined as

$$T_{\text{TEST}} = \{(b, d) \mid (b, d) \in T \wedge d \in D_{\text{TEST}}\}. \quad (4.4)$$

In Table 4.4 you can see the four cases which we can encounter while comparing the actual label b with the inferred labels.

Case	Actual Label	Inferred Label
(TP) True Positive	blocked	malicious
(FP) False Positive	allowed	malicious
(TN) True Negative	blocked	legitimate
(FN) False Negative	allowed	legitimate

Table 4.4: Definitions of outcomes of hypothesis testing

When used in text, by the FP we usually mean the overall number of all false positives in the whole dataset, and the same for the rest of the shortcuts. The absolute numbers are more or less meaningless, we usually seek some measures derived from them to evaluate the performance of the classifier [17]. You can see the basic measures in Table 4.5.

Name	Formula
(TPR) True Positive Rate	$(TP)/(TP + FN)$
(FPR) False Positive Rate	$(FP)/(FP + TN)$
Precision	$(TP)/(TP + FP)$
F-measure	$(2TP)/(2, TP + FP + FN)$

Table 4.5: Basic measures used to evaluate binary classifiers. The TPR and FPR are just relative values of TP and FP . The precision is a relative number of correctly labeled blocked connections. The F-Measure is a harmonic mean of the TPR and precision.

All of the measures depend on the choice of the threshold τ . The usual way to graphically display the performance is using Receiver Operating Characteristics (ROC) curve. The ROC curve is obtained by varying the τ over its all possible values and displaying the TPR as a function of FPR .

The proposed solution is intended to be used in anomaly detection system to find possible malicious domains among domains which are anomalous. The system presents its findings to a network administrator, and so we have to focus on keeping the number of false positives presented to the operator as low as possible. From our previous experience we want the false positive rate to be below 0.1%.

Focusing on maximizing of the TPR w.r.t. fixed FPR is not sufficient, because the overall precision might be still low. Maximizing only the precision is not sufficient also, because it might lead to low TPR . The TPR is important for the system to be even worth using, so we can not ignore it. The precision is also important, because with low precision the operator of the system has to deal with a large portion of presented false positives, so we can not ignore it also. To avoid the problem with multi-criteria optimization we use the F-measure, which averages those two metrics using harmonic mean.

4.4 Discussion of meaning of the parameters Θ

To evaluate the performance of the proposed algorithm we first need to optimize the parameters $\Theta = (u_a, v_a, u_b, v_b)$ of the Gamma prior. Because the $\Theta \in \mathbb{R}_+^4$ it is not an easy task to optimize it. Thus we need to first study the meaning of the parameters and check whether we can reduce the space of Θ .

From the definition of the expected value of Gamma distribution we know that the expected value of a, b is

$$a = \frac{u_a}{v_a}, \quad b = \frac{u_b}{v_b} \quad (4.5)$$

and that its variance is

$$\sigma_a^2 = \frac{u_a}{v_a^2}, \quad \sigma_b^2 = \frac{u_b}{v_b^2}, \quad (4.6)$$

which is valid for all a_k and b_k , but we will omit the lower index k for the sake of convenience. As discussed earlier a variation of a prior define its strength. We do not have any reason to a priori assume that the strength of the prior to a differ from the strength of the prior to b . To reduce the number of free variables we therefore set the two variances to be the same and introduce the strength $s_\gamma \in \mathbb{R}_+$ of the Gamma priors as

$$s_\gamma^{-1} = \sigma_a^2 = \sigma_b^2 \quad (4.7)$$

and use it to define the original parameters as

$$u_a = a^2 s_\gamma, \quad v_a = a s_\gamma, \quad (4.8)$$

and similarity for b .

This representation is much more pleasant to work with, because we understand the meaning of all the variables. On the other hand we still do not know how to set the desired value of a, b .

From the definition of the Beta prior we know that

$$m = \frac{a}{a+b}, \quad \sigma_m^2 = \frac{ab}{(a+b)^2(a+b+1)}, \quad (4.9)$$

where the value m is the overall expected maliciousness of a domain.

You can see that the variance of the m decreases with both a and b , and it is symmetric in both variables. Using this observation we can define the variables as

$$a = m s_\beta, \quad b = (1-m) s_\beta, \quad (4.10)$$

where the $s_\beta \in \mathbb{R}_+$ is the strength of the Beta prior. You can verify that if you substitute the (4.10) into (4.9) than the equality of mean values hold and the variance decreases with s_β .

The s_β could not be defined using the variance of Beta distribution, because we can not just select any value from \mathbb{R}_+ as the variance. The Beta distribution is defined on closed, unit length interval and therefore for each possible mean value there exists appropriate maximum value of the variance. Also the fact that the two strengths are defined differently does not matter because the variances of the distributions are not comparable anyway.

Parameter	m	s_γ	s_β	u_{a_0}	v_{a_0}	u_{b_0}	v_{b_0}
Value	0.6	0.08	5	0.72	0.24	0.32	0.16

Table 4.6: The chosen parameters of the model Θ

We have therefore replaced the original vector of parameters Θ by three new parameters σ, m, s as

$$u_a = m^2 s_\beta^2 s_\gamma \qquad v_a = m s_\beta s_\gamma, \qquad (4.11)$$

$$u_b = (1 - m)^2 s_\beta^2 s_\gamma \qquad v_b = (1 - m) s_\beta s_\gamma, \qquad (4.12)$$

which all have well defined meaning and therefore can be analyzed and optimized more easily.

4.5 Optimization of the parameters

In the previous section we have defined the original parameters Θ using a triplet of new variables with a well defined meaning. Because the space in which we need to find the optimal parameters is still three dimensional, it would be too computationally demanding to run a grid search to find the optimal settings. Also there is no way to optimize the parameters analytically. We therefore focus on optimizing all of the parameters separately, keeping the other parameters fixed.

In Figure 4.4 you can see the performance evaluated on the datasets from September and October. The graphs on the left were generated using the September dataset, the graphs on the right were generated using the October dataset. Graphs on each row were generated by varying one of the parameters but keeping the rest fixed.

At first we guessed the parameters to be $s_\beta = 5$, $s_\gamma = 0.04$, and $m = 0.5$. Then we optimized the parameter separately in the same order.

If you look at the graphs you will see that with increasing s_β the observed FPR changes inversely in the two datasets. Unfortunately this phenomenon is even more obvious during optimization of the prior maliciousness m . As you can see the optimal setting for the September dataset is the worst for October dataset and vice versa.

Our goal is to create a stable system with as low FPR as possible. By stable we mean that it would perform well over time. To prevent over-fitting we therefore focus on minimizing the maximum FPR between the datasets. By analysis of the measured values in Figure 4.4 we choose the settings described in Table 4.6.

4.6 Comparison with the Probabilistic Threat Propagation

To evaluate the quality of the inferred values we implemented the Probabilistic Threat Propagation algorithm (PTP) described in Section 2.1.3. The two approaches are hardly comparable, because the PTP needs as an input a list of malicious domains. On the other hand our algorithm infers the probability of maliciousness which can be then used to discriminate between malicious and legitimate domains automatically. This is why our goal is

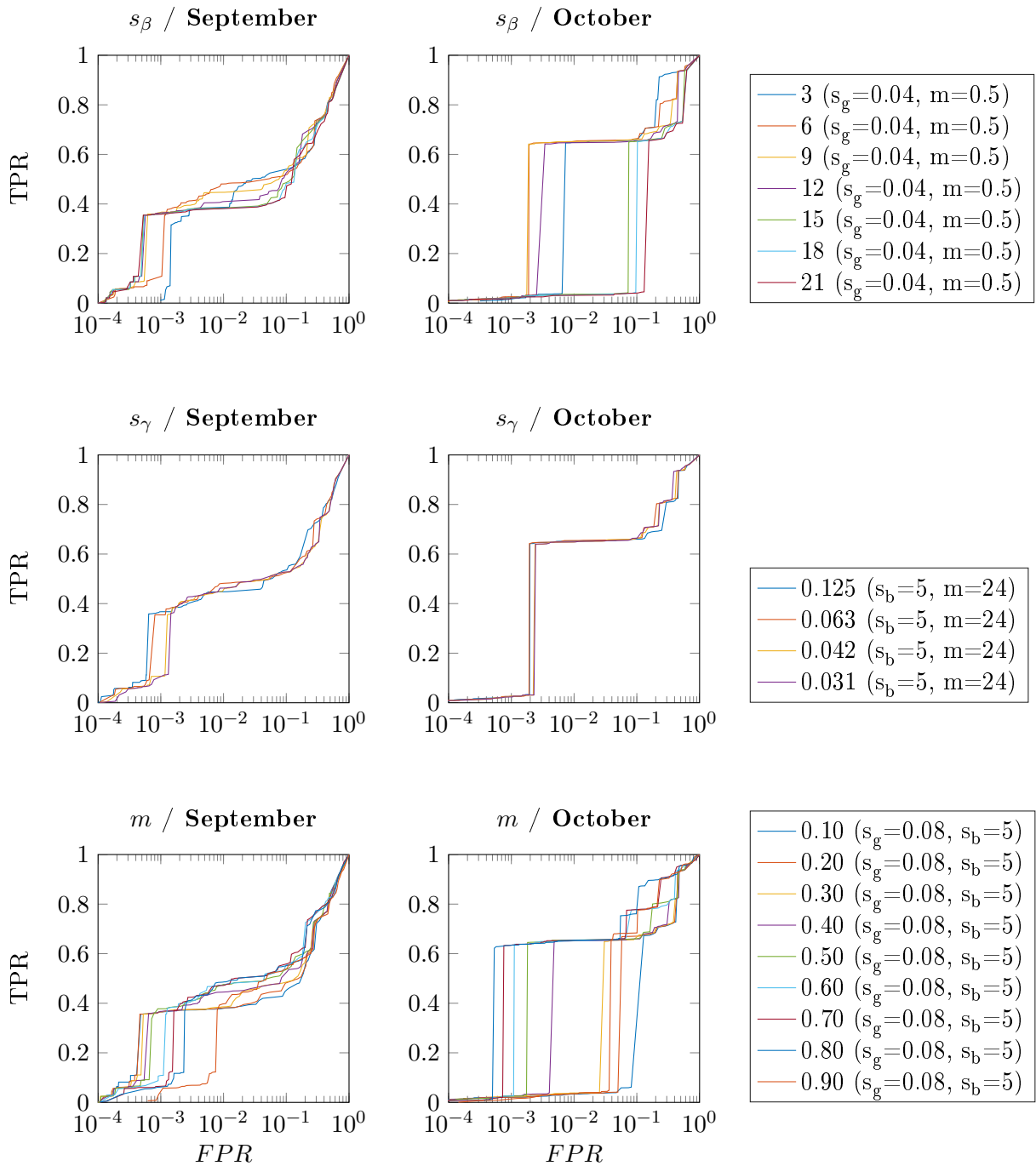


Figure 4.4: Optimization of the parameters s_β , s_γ , and m_d . Each of the rows shows optimization of one of the parameters, keeping the other parameters fixed. The columns represent evaluation of the parameters on different dataset. The classifiers on the left, and right were trained on September, and October dataset respectively and evaluated on actually observed evidence from following month.

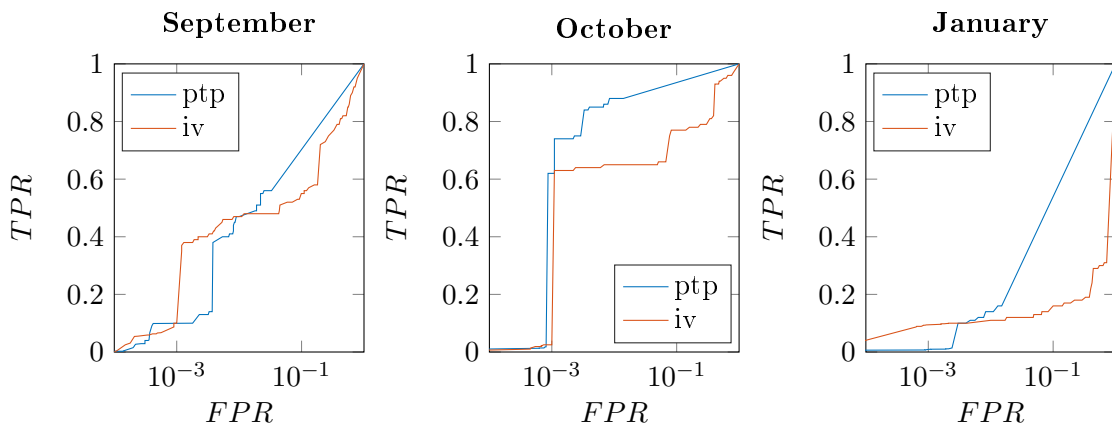


Figure 4.5: Comparison of the proposed algorithm with the Probabilistic Threat Propagation.

not to select the better performing of those two algorithms, but to analyse possible failure of our approach.

To get the list of malicious domains we created a rule of thumb. We consider a domain to be malicious, if the malicious ratio r_d , i.e. the relative number of blocked connections, is greater than 20%. We choose this threshold from our previous experience to filter nontrivial false positives, such as `yahoo.com`.

You can see the comparison in Figure 4.5. With the exception of the October dataset, our algorithm outperforms the PTP in the terms of FPR . On the other hand the PTP seems to be able to get to higher values of TPR . Interestingly both approaches failed on the January dataset, which might be caused by insufficient information contained in the WHOIS data.

In the Figure 4.6 you can see the evaluated metrics. Unfortunately the optimal threshold is not stable. This might be caused by the fact that the leaps in the precision and F-Measure are generated by a few malicious domain which had high traffic during the time of observation. We might just need even more data to find the optimal threshold on m_d .

4.7 Analysis of the inferred values

In this section we want to analyse how well does the model learn from the data and what is the impact of the Beta prior to the inferred values of m_d . We can answer those questions by analysing the correlation between the number of observed connections per domain N_d , the observed malicious ratio r_d , and the actually inferred value m_d .

The value m_d was defined using Bernoulli distribution as probability of observing blocked connection. Therefore if the observed samples are i.i.d., than with the N_d growing to infinity, the r_d should be equal to m_d . On the other hand if the N_d is low the expected value of m_d should be determined by the Beta prior.

To evaluate the described behaviour we plotted 2D histograms. On the horizontal axis there is a logarithm of the N_d and on the vertical axis there is a difference between m_d and

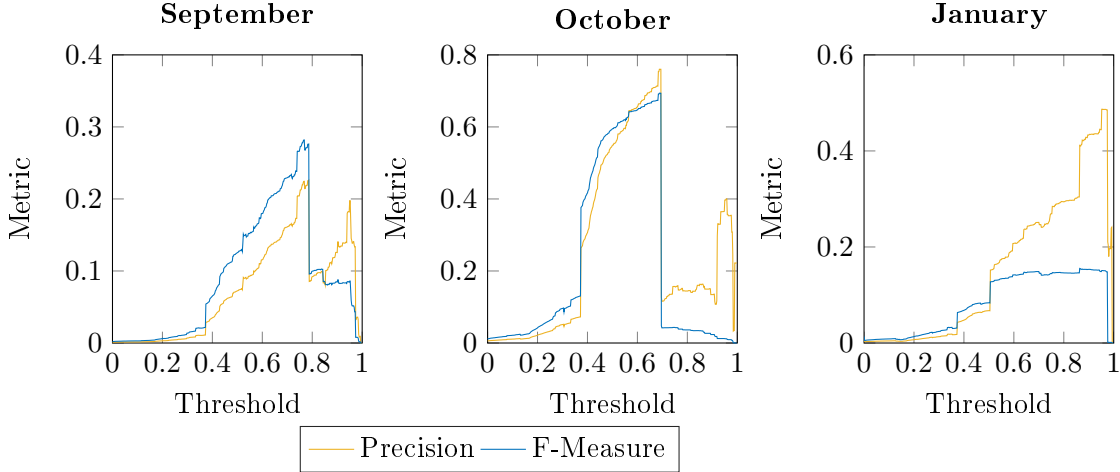


Figure 4.6: In this figure you can see the evaluated performance metrics on each of the available datasets. Please note that this is not the overall performance of the inference, but only the quality of the inferred prior. You can see that, for example, the precision ranges from as low as 15% to as much as 70%.

r_d . To learn how the prior affect the malicious and legitimate domains we treated those two groups of domains separately. For this analysis we considered domains to be potentially malicious if there was at least one block connection targeting the domain. We took the values of m_d from both testing and training data to see how the model learns from the given evidence and it generalizes.

In Figure 4.7 you can see the four resulting histograms. The brightness of the colours is proportional to logarithm of number of domains in the bins. The histograms on the left were generated using the evidence r_d , and N_d from the training data. It is clear that the model needs at least 100 samples to completely learn the behaviour of a domain. The values inferred from less than 100 samples are corrected by the conservative prior.

The histogram on the right were generated using evidence from testing data. The noise is probably caused by the high number of domains for which we have observed only a few connections in the following month. Overall the inferred values seem to match the evidence well. The important thing is that the values are clearly centralized around the line $m_d - r_d = 0$.

4.8 Overall performance

In the previous experiments we used the intersection of domains $D_{\text{TEST}} = U_{\text{PRED}} \cap O_{\text{SUCC}}$ to evaluate the performance on previously unseen domains. To get a general knowledge about the performance of the model as a whole, we need to evaluate it on the intersection

$$D'_{\text{TEST}} = D_{\text{PRED}} \cap O_{\text{SUCC}} = (O_{\text{PRED}} \cup U_{\text{PRED}}) \cap O_{\text{SUCC}}. \tag{4.13}$$

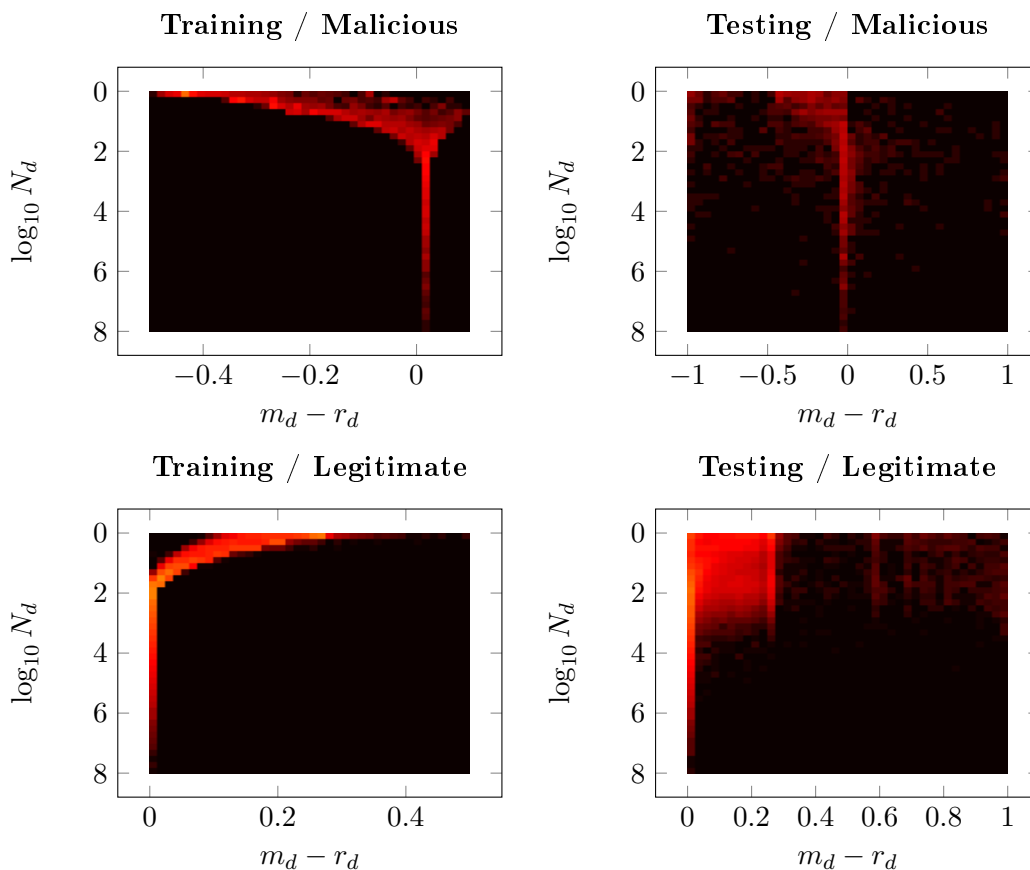


Figure 4.7: Histograms showing correlation between the number of connections to domain N_d and the difference between the inferred value m_d and actually observed relative number of blocked connections r_d . The graphs on the left get the value r_d from the training data. The graphs on the right get the value r_d from the testing data. The graphs on the top take into account only domains with $r_d > 0$. The graphs on the bottom take into account only domains with $r_d = 0$.

In Figure 4.8 you can see the evaluated performance. We chose the operating point according to the F-Measure, as you can see in Table 4.8. As we have already mentioned, it is because the F-Measure balances the two important measures; precision and TPR .

τ	TPR	FPR	Accuracy	Precision	F-Measure
0.498	0.522	0.000269	0.997	0.928	0.668

Table 4.7: Optimal operating point according to the F-Measure

Overall the performance is better than in the experiments with unseen domain only (see Figure 4.6), but it still could be possibly improved. This is due to the fact, that we do not distinguish sub-domains, as for example `a.example.com` and `b.example.com` from the `example.com` itself. Thus the actual evidence about the sub-domains gets mixed up. This

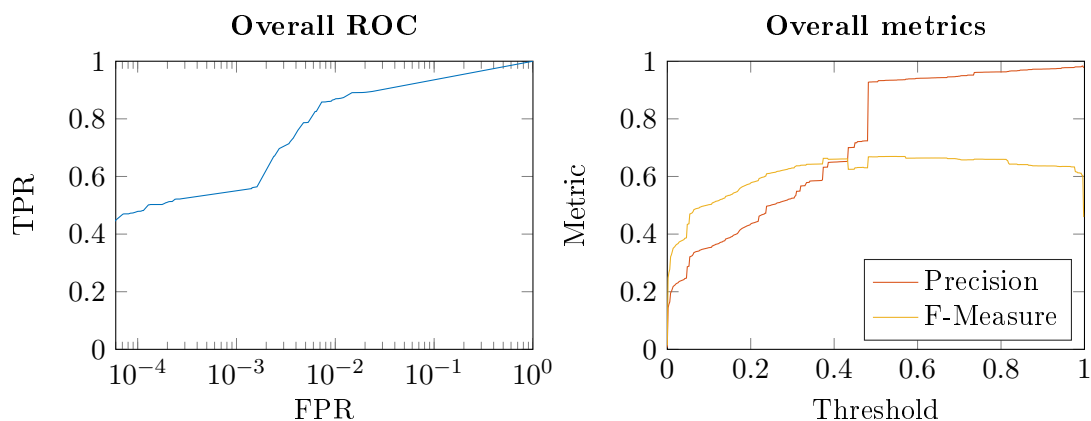


Figure 4.8: Overall performance of a linear classifier derived from m_d . The performance measures were evaluated on the intersection of all domains between the subsequent months. You can see that with the FPR equal to 1% we can get up to 90% TPR . Such FPR would be too high so with the FPR equal to 0.03% we can get 52% TPR and 93% precision.

causes that the sub-domains a and b are classified equally, which leads to false inference. We will tackle this problem in future research.

Chapter 5

Conclusion

In this thesis we have proposed a novel reputation system. The system builds a belief about observed domains based on their behaviour and is able to generalize to other, previously unobserved domains, based on information extracted from WHOIS records. It is build upon purely probabilistic model and uses the tools of Variational Bayes to learn its parameters.

In our experiments we were working with proxy-logs from a major Intrusion Prevention System (IPS). In each log, we were interested in the domain name from the URL and the flag whether the connection was blocked. The system uses this data, together with keys extracted from the WHOIS records, to build a belief about maliciousness of domains occurring on the internet.

The form of the available data allowed us to naturally create training and testing sets from subsequent time windows. Using the true evidence from the future instead of splitting the labeled data into training and testing sets brings a noise to the data but it is able to capture the true predictiveness of the system.

During the evaluation we have analyzed how the inferred probability of maliciousness would perform as a feature. We found that with reasonable false positive rate, around 0.03%, the system has 52% true positive rate and 92% precision.

In similar manner we also evaluated the how the inferred prior, not taking into account any direct evidence, performs as a feature. The both precision and true positive rate ranged from 20% to 70% on different datasets with false positive rate as low as 0.1%.

We found out that with more than 100 observations of a domain the system is able to completely learn its behaviour. If less than 100 samples is present, the prior influence the belief about the domain.

The system is currently designed for off-line learning, but as the model is composed of conjugate pairs of distributions it can be easily extended for continual on-line learning. Also from historical reasons we took into account only the level of domain name following the public suffix, ignoring the sub-domain. Future experiments can be done to show the impact of distinguishing different sub-domains on the precision of the system.

Bibliography

- [1] A closer look: Perkele android malware kit, 2013. Available online at <http://krebsonsecurity.com/tag/sms-malware/>.
- [2] Threat spotlight: A string of ‘paerls’, part 2, deep dive, 7 2014. Available online at <http://vrt-blog.snort.org/2014/07/threat-spotlight-string-of-paerls-part.html>.
- [3] Threat spotlight: A string of ‘paerls’, part one, 6 2014. Available online at <http://blogs.cisco.com/security/a-string-of-paerls>.
- [4] Android banking trojan and sms stealer floating in the wild, 1 2015. Available online at <http://research.zscaler.com/2015/02/android-banking-trojan-and-sms-stealer.html>.
- [5] Icann - history of whois, 4 2015. Available online at <http://whois.icann.org/en/history-whois>.
- [6] Michael Bailey, Jon Oberheide, Jon Andersen, Z Morley Mao, Farnam Jahanian, and Jose Nazario. Automated classification and analysis of internet malware. In *Recent advances in intrusion detection*, pages 178–197. Springer, 2007.
- [7] Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi. Exposure: Finding malicious domains using passive dns analysis. In *NDSS*, 2011.
- [8] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
- [9] Kevin M Carter, Nwokedi Idika, and William W Streilein. Probabilistic threat propagation for malicious activity detection. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 2940–2944. IEEE, 2013.
- [10] Bled Electronic Commerce, Audun Jøsang, and Roslan Ismail. The beta reputation system. In *In Proceedings of the 15th Bled Electronic Commerce Conference*, 2002.
- [11] L. Daigle. Whois protocol specification, 9 2004. RFC 3912.
- [12] Mark Felegyhazi, Christian Kreibich, and Vern Paxson. On the potential of proactive domain blacklisting. In *Proceedings of the 3rd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*, pages 6–6. USENIX Association, 2010.

- [13] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
- [14] Nektarios Leontiadis and Nicolas Christin. Empirically measuring whois misuse. In *Computer Security-ESORICS 2014*, pages 19–36. Springer, 2014.
- [15] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1245–1254. ACM, 2009.
- [16] Zhanyu Ma and Arne Leijon. Bayesian estimation of beta mixture models with variational inference. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(11):2160–2173, 2011.
- [17] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.
- [18] Andrew G West and Aziz Mohaisen. Metadata-driven threat classification of network endpoints appearing in malware. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 152–171. Springer, 2014.
- [19] Wei Xu, Kyle Sanders, and Yanxin Zhang. We know it before you do: Predicting malicious domains. 2014.

Appendix A

Variational Bayes method

Let us assume that we have a complete Bayesian model $p(\mathbf{z}, \mathbf{x})$, where the \mathbf{z} are parameters of the model, and the \mathbf{x} are the observed data. Also let us assume that the distribution is not tractable, because it can not be analytically evaluated. The Variational Bayes seeks an approximation $q(\mathbf{z})$ to the posterior distribution $p(\mathbf{z}|\mathbf{x})$.

The derivation proceeds as follows

$$\log p(\mathbf{x}) = \log p(\mathbf{z}, \mathbf{x}) - \log p(\mathbf{z}|\mathbf{x}) \quad (\text{A.1})$$

$$= \int_{z \in Z} q(\mathbf{z}) [\log p(\mathbf{z}, \mathbf{x}) - \log p(\mathbf{z}|\mathbf{x})] dz \quad (\text{A.2})$$

$$= \int_{z \in Z} q(\mathbf{z}) \log \frac{p(\mathbf{z}, \mathbf{x})}{q(\mathbf{z})} dz - \int_{z \in Z} q(\mathbf{z}) \log \frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z})} dz \quad (\text{A.3})$$

$$= \mathcal{L}(q) + \text{KL}(q||p), \quad (\text{A.4})$$

where the $\text{KL}(q||p)$ is KL divergence between the distributions q and p which is in general defined as

$$\text{KL}(q||p) = \int_{z \in Z} q(\mathbf{z}) \log \frac{p(\mathbf{z})}{q(\mathbf{z})} dz \quad (\text{A.5})$$

and it is equal to the number of extra bits needed to compress an information channel described by probability distribution q if we use an algorithm optimized for p . We will state without a prove that $\text{KL}(q||p) \geq 0$ and that $\text{KL}(q||p) = 0$ if, and only if, the p and q are the same. Therefore the $\mathcal{L}(q)$ is a lower-bound to the $\log p(\mathbf{x})$ because it holds that

$$\mathcal{L}(q) \leq \log p(\mathbf{x}) \quad (\text{A.6})$$

and it is maximized by minimizing the KL divergence between the $q(\mathbf{z})$ and $p(\mathbf{z}|\mathbf{x})$.

Now let us have a space of tractable probability distributions \mathcal{Q} . To approximate the posterior distribution $p(\mathbf{z}|\mathbf{x})$ by a tractable distribution $q(\mathbf{z})$, where $q \in \mathcal{Q}$, we must maximize the lower-bound $\mathcal{L}(q)$.

The approximation problem can be interpreted as an optimization problem of finding such $q \in \mathcal{Q}$ for which hold that the $\text{KL}(q||p)$ is minimal. It just depends on the way how we specify the set of tractable distributions \mathcal{Q} .

A.1 Factorized distributions

One way to select the space \mathcal{Q} is by assuming that the vector \mathbf{z} can be partitioned into N disjoint sets as $\mathbf{z} = (\mathbf{z}'_1, \dots, \mathbf{z}'_N)$. Then the \mathcal{Q} is defined as

$$\mathcal{Q} = \left\{ q \mid q(\mathbf{z}) = \prod_{i=1}^N q_i(\mathbf{z}'_i) \right\}, \quad (\text{A.7})$$

which is a space of all distributions over \mathbf{z} which are defined as products of its marginals. This definition does not require the $q \in \mathcal{Q}$ to be from any specific family of distributions, or to be parametrized by any specific parameter. The partitioning is done to simplify the intractable distribution and it is selected in advance. Hence we must be careful while selecting the partitions, because from the (A.7) it holds that each two variables which are not in the same partition are forced to be independent.

By plugging the (A.7) into the (A.3) we get

$$\mathcal{L}(q) = \int q(\mathbf{z}) \log \frac{p(\mathbf{z}, \mathbf{x})}{q(\mathbf{z})} d\mathbf{z} \quad (\text{A.8})$$

$$= \int \prod_{i=1}^N q_i(\mathbf{z}'_i) \left[\log p(\mathbf{z}, \mathbf{x}) - \sum_{i=1}^N \log q_i(\mathbf{z}'_i) \right] d\mathbf{z}, \quad (\text{A.9})$$

where we just replaced both occurrences of $q(\mathbf{z})$ by the product of its marginals.

Now because all of the marginals q_i are independent we can focus on optimizing each of them separately by assuming all q_j such that $j \neq i$ to be fixed. It was shown in [8] that by rearrangement of the (A.9) we can obtain

$$\mathcal{L}(q) = \int q_j(\mathbf{z}_j) \left[\int \log p(\mathbf{z}, \mathbf{x}) \prod_{i \neq j} q_i(\mathbf{z}_i) d\mathbf{z}_{i \neq j} \right] d\mathbf{z}_j - \int q_j(\mathbf{z}_j) \log q_j(\mathbf{z}_j) d\mathbf{z}_j + \text{cons.} \quad (\text{A.10})$$

$$= \int q_j(\mathbf{z}_j) \mathbb{E}_{i \neq j} [\log p(\mathbf{z}, \mathbf{x})] d\mathbf{z}_j - \int q_j(\mathbf{z}_j) \log q_j(\mathbf{z}_j) d\mathbf{z}_j + \text{cons.} \quad (\text{A.11})$$

$$= \int q_j(\mathbf{z}_j) \log \frac{\tilde{p}(\mathbf{z}, \mathbf{x})}{q_j(\mathbf{z}_j)} d\mathbf{z}_j + \text{cons.}, \quad (\text{A.12})$$

where the distribution $\tilde{p}(\mathbf{z}, \mathbf{x})$ is defined as

$$\log \tilde{p}(\mathbf{z}, \mathbf{x}) = \mathbb{E}_{i \neq j} [\log p(\mathbf{z}, \mathbf{x})]. \quad (\text{A.13})$$

You can see that the (A.12) has a form of negative KL divergence. To maximize the $\mathcal{L}(q)$ we thus need to minimize it. The minimum is obtained if the distributions $\tilde{p}(\mathbf{z}, \mathbf{x})$ and $q_j(\mathbf{z}_j)$ are equal and therefore we set the q_j to be

$$\log q_j(\mathbf{z}_j) = \mathbb{E}_{i \neq j} [\log p(\mathbf{z}, \mathbf{x})]. \quad (\text{A.14})$$

Because the KL divergence is minimized by recalculating the i -th marginal, the $\mathcal{L}(q)$ is guaranteed to be maximized. We can therefore reach an optimal value of $\mathcal{L}(q)$ in an iterative manner by successive recalculating of the q_i . On the other hand we do not have any guarantees about uniqueness of the found solution, and therefore its quality might depend on the initial estimates of the parameters of the marginals.

Appendix B

Nomenclature

CWS Cloud Web Security

EM Expectation Maximization

HDFS Hadoop Distributed File System

ICANN Internet Corporation for Assigned Names and Numbers

IPS Intrusion Prevention System

KL Kullback–Leibler

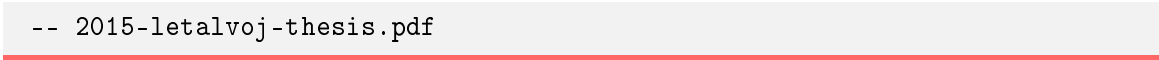
ML Maximum Likelihood

ROC Receiver Operating Characteristics

URL Uniform resource locator

Appendix C

CD content



```
-- 2015-letalvoj-thesis.pdf
```

Figure C.1: Content of the attached CD