ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK Institut für Technische Informatik und Kommunikationsnetze

Kirila Adamova

Anomaly Detection with Virtual Service Migration in Cloud Infrastructures

Master Thesis 263-0800-00L
October 2012 to March 2013

Tutor: Dr. Paul Smith
Co-Tutor: Dominik Schatzmann
Supervisor: Prof. Bernhard Plattner

Acknowledgements

I would like to thank everybody who has supported me during this thesis work.

First and foremost I would like to thank Prof. Bernhard Plattner, first, for igniting my interest in the field of Network Security and for giving me the opportunity to work in his group.

I would also like to thank Paul Smith, who has been with me every step of the way. His positive attitude has provided an important source of motivation and his availability and constant feedback were very much appreciated.

During this thesis I had to employ and extend existing tools, and I want to thank their developers for all the help. Thanks to Dominik Schatzmann for spending hours of his time to explain how FlowBox works, helping with flow-reading scripts, sharing ideas and challenging decisions about my project. Furthermore, I would like to also thank Bernhard Tellenbach for providing and explaining the anomaly detection tool and Flame. I am especially grateful to him for replying to my endless emails and spending time preparing the tools for my needs.

I also want to thank my parents for making it possible for me to come to study in Switzerland.

Finally, I would like to thank D-HEST ISG for all the cake.

Abstract

Cloud computing is becoming a very popular way to outsource data storage, services or even infrastructure. Clouds are large network data centers that make use of virtualization technology, thus enabling dynamic scalability and , from user's perspective, apparent infinite resources. Clouds host services in virtual machines that are isolated from one another and can be migrated within or between data centers. Virtual service migration is used in case of failure or disaster, resource optimization issues, maintenance and to reduce network costs. This thesis work shows how virtual service migration affects state-of-the-art anomaly detection that can be used to detect malicious activity, such as Distributed Denial of Service attacks. Using data from actual services, anomalous traffic is analyzed with and without migration in different scenarios. The scenarios cover different types of anomalies, i.e. attacks, variations of anomaly intensity and variations of size of migrated services. All these parameters affect how much virtual service migration influences anomaly detection results. Our results show that, in some cases, virtual service migration can be incorrectly detected as an anomaly, i.e., an attack. To mitigate this, one could adjust detection parameters such as the anomaly score threshold to reduce the number of incorrectly detected attacks. However, this introduced the risk of not detecting genuine attacks. This finding brings into question the reliability of state-of-the-art anomaly detection techniques when applied in the cloud, pointing to the need for further investigation in this important area.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Cloud

The cloud is an emerging technology that shifts infrastructure to networked data centers to reduce costs of hardware and software resource management [51]. It is popular among enterprises and individuals who are looking for more resources for their needs because of its instantly and infinitely scalable infrastructure [51]. The cloud attracts with immediate scalability, optimization of resource usage and virtualization [51]. Everything it provides is delivered as a service over a network. Some of the most popular services a client can purchase are Infrastructure as a Service (includes virtual machines, storage, servers, network resources), Software as a Service (the application software is hosted on the cloud and users access it through cloud based clients), Platform as a Service (the cloud provides a computing environment), Storage as a Service (data storage), Network as a Service (inter-cloud network connectivity), and Security as a Service (security management). Current examples of usage of clouds include storing documents, outsourcing enterprise computing in the cloud or providing infrastructure [29].

Clouds have been defined in various ways. According to [51], clouds are a large pool of easily usable and accessible virtualized resources. These resources can be dynamically reconfigured to adjust to a variable load, allowing also for optimum resource utilization. The minimum cloud definition includes scalability, pay-per-use utility model and virtualization. [49] presents the opinions of twenty one experts on clouds and cloud computing. They define a cloud as an infrastructure which can be scaled on demand within minutes or even seconds and optimizes utilization of resources. Another view from the same article is that a cloud is an Internet-centric software which is scalable, multi-tenant, multi-platform, multi-network, and global, or it is web-based services with a wide range of functional capabilities. Clouds are virtualized hardware environments, where application infrastructure is easily configured, deployed, dynamically scaled and managed. Another expert says that a cloud has access to resources and services needed to perform functions with dynamically changing needs. It is a virtualization of resources that maintains and manages itself. More definitions continue to mention on-demand resource allocation, virtualization of resources, outsourcing services, flexibility, reliability, scalability, security, hiding complexity and extending existing IT capabilities. According to [29], among the benefits of cloud computing is outsourcing computation, which eliminates many compatibility issues. In addition, updates and bug fixes are deployed withinin minutes. All these definitions give different viewpoints of clouds, but they all present several common characteristics - scalability, virtualization, dynamic allocation and optimization of resources.

The cloud concept is based on virtualization, utility computing and distributed computing [51]. In order to have the capability to provide outsourcing services and unlimited resources the cloud uses several core technologies. The key elements of cloud computing are immediate scalability, resources usage optimization, utility computing, user-friendliness, massive data scalability and virtualization [51]. These characteristics are provided by monitoring and automation of resources management, by a dynamic environment, a data center comprised of a huge collection of clusters, load balancing and data and process outsourcing [51]. The use of resources is op-

timized such that logically, the number of resources appears to be unlimited. Virtualization of services and of resources allows for the dynamic allocation of more resources. It also provides security by its main principle. [8] says that before virtualization was introduced, placing different customers' workloads on the same physical machines could lead to security vulnerabilities and possible loss of sensitive data. The proposed implementation uniformly enforces an isolation policy. It specifies which virtual machines can access which resources and with whom they can communicate. Customer workloads are isolated from each other, which prevents leaking data, ensures viruses cannot spread and prevents misconfiguration [8]. Another benefit of virtualizing services is the freedom of moving them between physical resources because they are not bound to any of them. On the logical level no changes occur. This is called virtual service migration and is executed, for example, in cases of maintenace, device failure, or the need for load balancing/physical optimization.

1.2 Motivation

Many organizations have been looking to move their services and business processes to the cloud [36]. However, as more of them consider the cloud for critical business applications, they realize the challenge of combining existing internal controls with cloud protection [20]. The cloud gives a lot of benefits, but there are still challenges to be solved [48]. According to [37], security was and will always be a critical factor in cloud-computing decisions. Consequently, securing cloud infrastructures is of paramount importance. Virtualization already provides internal security between services that are not allowed to access each other, but it does not take care of attacks coming from outside the cloud. The cloud provides many services, one of which is Security as a Service, and this can be implemented individually in each virtual machine that has requested security services. Such an approach can detect attacks with a dominant destination such as Distributed Denial of Service (DDoS) attack, but will miss wide-spread attacks such as port scans. For the latter type to be detected, a larger portion of the data center traffic needs to be analyzed. Moreover, supporting security mechanisms for each individual virtual machine introduces significant undesirable management overhead. In addition, attacks will not only target particular services but the infrastructure and resources of the cloud data center as well. There are multiple protection mechanisms that can be adopted to make cloud infrastructures more secure such as firewalls and intrusion detection systems. However, the cloud introduces new techniques such as virtualization and service migration within a datacenter or even among different geographical locations. It is not yet clear what effect those techniques have on the existing security mechanisms. The motivation behind this work is to determine how virtual service migration would affect results from anomaly detection systems.

1.3 Project Goal

The main task of this thesis is to find out how virtual service migration in clouds affects state-of-the-art anomaly detection techniques. This includes researching cloud infrastructure and traffic, and possible attacks on it, as well as finding out if and which existing anomaly detection techniques can be applied to clouds. Afterwards, it will be determined where in the cloud structure the appropriate detection tools can be used. This project focuses on how specific cloud characteristics such as virtualization and migration work, how they change traffic data and affect results from existing anomaly detection techniques. Finally, the theory will require empirical testing. Methodology steps are followed to simulate cloud traffic behavior, inject anomalies, simulate migration and observe how anomaly scores change when migration is involved. Finally, a conclusion will be made about migration's affect on anomaly detection in clouds.

1.4 Overview

Chapter 2 presents background information on existing research in the fields of network data centers, data center traffic, traffic analysis, anomaly detection and virtual service migration. This research is necessary to understand decisions taken throughout the course of this work. A lot

of these decisions are involved in the methodology described in Chapter 3. It explains how a typical cloud system is simulated and provides detailed steps about preparing and executing experiments. These steps include processing traffic data by writing extensions for existing tools, which are also included in this chapter with short descriptions of their functionality. Once the backbone of the experiments is established, Chapter 4 provides details on the exact experiment scenarios that were conducted. It presents the analyzed anomalies, the traffic traces used and the migration applied to those traces. The combination of those parameters forms the experiments. The results of these experiments are shown in Chapter 5. These results are illustrated using plots and analyzed based on their behavior. Chapter 6 completes this work by summarizing all the decisions taken during this project and the key findings based on the presented results. Furthermore, it includes suggestions for future work and improvements.

Chapter 2

Background

The background section introduces major topics related to this project that will help the reader understand decisions taken throughout it. This work involves the cloud infrastructure and traffic which hold their bases in network data centers. First, data center architecture is explored by providing details about its structure, addressing and routing. The chapter continues with data center traffic, its complexity and behavior. The next section discusses aspects of traffic, focusing on the location of analyzing traffic in the infrastructure and how different data processing affects the traffic. Traffic analysis is done to detect anomalies, and the chapter continues by presenting types of anomalies and different anomaly detection techniques. The last section of this chapter introduces a new aspect in the data center common for cloud data centers - virtualization and virtual service migration. The migration process and the changes it causes to traffic behavior are presented in that section.

2.1 Network Data Centers

Clouds in their core are larger network data centers which provide scalability and use virtualization. Today they host between 50,000 and 200,000 servers [27]. This section describes various network data center architectures by focusing on the typical cloud data center infrastructure. It also discusses internal and external traffic behavior in that infrastructure.

2.1.1 Architecture

Infrastructure

According to [1, 4, 6, 21, 26, 27, 50], typical data center architectures of today consist of two- or three-level trees of switches or routers. A three-tiered design has a core tier in the root of the tree, an aggregation tier in the middle and an edge tier at the leaves of the tree. For a visual reference, please see Figure 2.1 below. A two-tiered architecture has only the core and the edge tiers, supporting smaller infrastructures [1]. A similar current data center architecture is briefly discussed in [21]. For their product they propose an architecture of 2-level multi-rooted tree of pod switches and core switches, where the core switches are either electrical packet switches or optical circuit switches. Another infrastructure example, described in [50], captures a 1,000-machine deployment by having 25 racks with each rack having up to 40 end-host machines. Each end-host connects to a top-of-rack (ToR) switch, in a similar way to [4], where the infrastructure consists of virtual machines (VMs) hosted on servers connected to ToR switches. These are then connected to Pod switches which in turn are connected to Root switches, in a three-level tree manner. A similar design by [26] describes a hierarchy starting at the bottom with a layer of servers in racks, typically 20 to 40 servers per rack, each of them connected to a ToR switch. Then each of these ToR switches is connected to two aggregation switches for redundancy, and these switches are connected to the next level to access routers. At the top of the hierarchy, core routers carry traffic between access routers and manage traffic into and out of the data center [26].
The reasoning behind these architectures lies in solving the scalability bottleneck issue in large-scale clusters [1]. According to [38], most existing data center architectures are expensive to

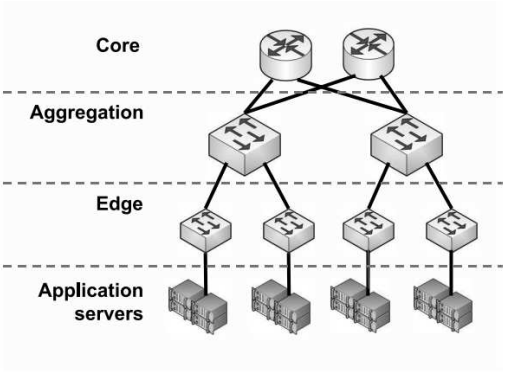


Figure 2.1: A 3-tier data center topology [6]

scale, provide limited support for multi-tenancy and require complex configuration. NetLord [38] proposes a solution for a multi-tenant virtualized cloud data center, which scales at low cost, is easy to configure and operate and has a flexible network abstraction. One of the problems in existing data center infrastructures is the limited space in routing tables in commodity switches. This creates a scaling problem for MAC address learning, because switches cannot be exposed to the MAC addresses of all tenants, and a rapid arrival rate of new addresses can lead to flooding. This problem is solved by using commodity switches, which only store the addresses of other switches and not the addresses of virtual machines [38]. The switch configuration is static and does not need to change unless the network topology is modified. [25] confirms that commodity switches such as ToRs and aggregation switches are highly reliable. Similarly, a proposed solution by [1] suggests using lower cost commodity switches that can store routing tables to deliver scalable bandwidth. Those switches allow all directly connected hosts to communicate with one another at the full speed of their network interface. They have the capacity to aggregate and transfer packets between leaf switches. The composed infrastructure follows a special instance of Clos topology called fat-tree, interconnecting those commodity Ethernet switches [1, 26, 27, 41]. [50] also describes a fat-tree topology typical of data center networks. A fat tree network has links that become "fatter", i.e. have higher bandwidth as one approaches the root of the tree. Research from [4] also discusses a fat-tree topology that offers multiple paths between physical machines. They also suggest a virtual network connecting tenants' compute instances. A scalability solution is also proposed by [26], where they scale out the devices to build a broad network with a huge aggregate capacity using many inexpensive devices. The topology is another variation of a Clos network, where the links between the aggregation and core switches form a bipartite graph. The solution proposed by [26] uses multiple core switches and fewer faster links, whereas a fat-tree topology uses multiple core switches and larger amount of lower speed links [41]. Another possible topology is BCube, which is a hypercube-like topology that uses the hosts to relay traffic [41]. A different approach is offered by [30], who discuss the idea of including wired or wireless flyways, extra links between pair of ToRs, in order to eliminate oversubscription.

Addressing and Routing

The next part of a data center architecture is the addressing of the services hosted there and of the switches directing the traffic. There are a number of ways this addressing is currently implemented. VL2's [26] addressing scheme separates server names from locations by using application-specific addresses (AAs) and location-specific addresses (LAs) respectively. VL2 uses a directory system that stores the mapping between names and locations. The reason for separating names from locations is to provide agility, i.e. support for hosting any service on any server, rapid growing and shrinking of server pools and rapid virtual machine migration. Switches are assigned LAs, and they run a layer-3 link-state routing protocol, which allows them to obtain the complete switch-level topology. Applications use AAs which remain unaltered despite changes of server locations due to virtual-machine migration. Each AA is associated with an LA, the identifier of the ToR switch to which the server is connected. Servers are configured to believe they all belong to the same IP subnet. When a packet is sent, the source

server generates an Address Resolution Protocol (ARP) request, which is intercepted by the VL2 agent, which asks the directory system for the correct mapping of AA to LA. Then the packet is encapsulated with the LA of the source ToR switch and sent to the LA of the destination ToR switch. When the packet arrives, it is decapsulated and forwarded to the respective server.

The NetLord design [38] gives a similar proposal for addressing. It suggests to avoid network-related restrictions in the placement of virtual machines and to build a data center network that fully virtualizes the layer 2 and 3 address spaces for each tenant and uses packet encapsulation. An example of an existing data center that follows these rules is Amazon, which provides a layer 3 abstraction and full IP adress space virtualization. Following NetLord's design, a packet is sent in the following way. First, a virtual machine needs to send a packet out. The layer 2 outgoing packet arrives at the local NetLord Agent. Using a NetLord look-up the destination edge-switch MAC address and server port number are determined. Then, a path for the packet is selected. Subsequently, it is encapsulated by the NetLord Agent, using the just determined layer 3 destination address for encapsulation that exploits features of both layers. This allows the edge switch to deliver the packet to the correct server and in turn also allows the destination NetLord Agent to deliver the packet to the correct tenant. Due to the separation in layers, the tenant virtual machine addresses are not known by the hardware switches, which only need to store addresses of other switches.

2.1.2 Traffic

Data center traffic is composed of internal and external traffic. Internal traffic accounts for server to server requests (where both servers are hosted on the same data center), replication of existing servers, and more. Datacenter networks carry background traffic consisting of long-lived flows, which involve large data transfers [50]. In most cases 80% of overall traffic volume is internal and the trend is towards even more internal communication [26, 27]. It includes bulk, network files, backup, data mining, index computations, MapReduce and back end to front end [27, 31, 40],. According to [40], for enterprise data center traffic it is also true that 80% of the flows are distributed within the enterprise. An example of internal traffic is virtual machines belonging to the same client communicating over the network shared between all clients. Thus, the bandwidth achieved by traffic between VMs depends on variety of factors such as network load and placement of those VMs [4]. External traffic is defined as flows that come from or are sent to somewhere outside the current data center. What is representative of external traffic in cloud data centers is that requests arrive over time and are not a static set of jobs [4].

Traffic routing was briefly discussed previously, but there are other aspects besides addressing that need to be discussed. It has to be coordinated in such a way to avoid bottlenecked links. Some implement load balancing, as presented in [26], that causes traffic between any pair of servers to bounce off a randomly chosen switch in the top level. Others do not rely on randomization for load balancing but use hash-based implementations [4]. Traffic is split per flow across multiple links. For a uniform distribution, a centralized controller can be used to reassign flows or distribute traffic based on packets. According to [41], the simplest solution is randomized load balancing where each flow is assigned a random path from a set of possible paths. However, random selection causes hot spots to develop, hence, they propose a different solution - to split flows based on a hash of the five tuple in each packet.

Another aspect of data center traffic is its behavior. Routing determines the path of flows only once the destination is known. [27] says that traffic patterns change nearly constantly, which makes them complex to model. Focusing on mining data centers, [31] defines some patterns about internal communication - a server either talks to almost all other servers in its rack or less than 25% of servers in its rack, and a server talks up to 10% to servers outside its rack. [31] also claims that a cloud data center, different from a mining data center, primarily deals with generating responses for web requests and will have different characteristics. Another paper discussing internal traffic behavior, [6], obtains conclusions based on data from five cloud data centers and compares it to traffic data from two private enterprise data centers and

three university campus data centers. It states that in cloud data centers a majority of traffic originating from the servers (80%) stays within the rack, whereas for university and private data centers most of the traffic leaves the rack. Another difference between a cloud data center and a private or university one is the amount of servers and devices in the data center. The cloud data centers, which are part of the research in [6], have 5 times more servers and devices than a private data center, and 10 to 15 times the servers and devices of a university data center. These cloud data centers provide support for wide range of services such as MapReduce, messaging, webmail, and web portal. Each of these applications consists of dependencies deployed across the data center, and this application mix impacts the traffic results.

The goal of the research in [7] is to study traffic patterns in data center networks. They discuss how traffic on aggregation and edge links is more bursty compared to traffic on the core links. This is a consequence of greater degree of losses observed near the edges. Core links show less packet loss but, on the other hand, they are more heavily utilized.

2.2 Traffic Analysis

The purpose of traffic analysis is to detect network traffic anomalies that could disrupt the normal operation of the network infrastructure. According to [12], anomaly detection or analysis typically operates on preprocessed traffic. This section shows different approaches on how and where traffic is analyzed.

According to [40], traffic analysis is conducted at the transport level. Scanning traffic is removed before proceeding with analysis because it causes the magnitude of traffic to increase. In [28], they analyze traffic collected at the endhosts. In [45], traffic is also filtered at the border of the network. [33] says that analysis is done at the entry point of the data center.

In [7] they have coarse-grained data for their experiments, such as average traffic volumes and loss rates. One of the data sets they use is collected from 19 data centers. To benefit from the coarse-grained data, they aggregate traffic statistics in 5 minute increments. Also they approximate traffic at the edge switches by generating fine-grained data from the coarse-grained one.

In contrast, traffic analysis in [45] is done on a huge amount of data. Their goal is to evaluate the effect of traffic mix and packet sampling on anomaly visibility. Traffic mix means that data is collected from different border routers. Packet sampling is reducing the amount of traffic data measured. Sampling at rate n means that every n^{th} packet from the original traffic data is inspected uniformaly at random and its features are recorded. The purpose is to summarize the traffic data. However, sampled traffic is incomplete, because small flows are likely to be missed. This could affect count metrics, but the impact of packet sampling on entropy and volume metrics depends largely on the traffic mix and type of anomaly. For example, flow counts are not a suitable metric for detecting flow-based anomalies, such as the Blaster worm, when packet sampling is used [13]. The authors of [45] conclude that while simple feature counts are well-suited for detection of anomalies that cause widening of the port or IP distribution, entropy metrics are more appropriate for detecting anomalies that cause a concentration of the distribution on a specific port or IP address.

Packet sampling can also be done randomly by selecting each packet with a certain probability [12, 13]. Each packet is selected with a probability q or discarded with probability $1 - q$. It reduces the amount of data, but its effects have to be assessed before using sampled data. Probability based packet sampling can disturb flow counts as well as the uniformly distributed sampling because small flows are sampled with a lower probability than larger flows. Packet counts are not affected by sampling. Entropy is also more robust to sampling that flow counts [13]. Packets are preprocessed by sampling and temporal aggregation [12]. The latter means that all packets in a measurement interval are represented by their temporal mean. The goal is to transform the traffic trace to a desired granularity before it is observed for anomalies. Results in [12] show that with aggregation the noise level increases and so does the false negative probability. However, if aggregation is replaced by a low-pass filter to obtain a better

spectrum estimation, the false negative probability improves. Once the traffic data has been preprocessed, the results are stored in timeseries, which are entropy reduced by anomaly detection and then a threshold is applied. The result is a series of alerts. Entropy reduction is the process of filtering the normal behavior from the time series. [12] concludes that the main performance criteria is defined as the likelihood that anomaly detected in raw data is also detected in the processed data with the same threshold.

A different idea on how to decrease amount of traffic analyzed is proposed by [34]. They suggest not analyzing all traffic in the data center but instead have switches and routers detect if there is a change in network behavior at their end, and use traffic from devices that have detected abnormal behavior. Then all this traffic will be aggregated and analyzed together. Another method for traffic analysis is to approach it as proposed in [16], and record packet sequences that match specified application-specific signatures.

2.3 Anomaly Detection

The first step of analyzing traffic is to establish a baseline of normal traffic. One way to create a baseline is to use the average over a past time period [13]. Any deviations from this baseline behavior are called anomalies. Anomaly size is defined as the distance between sample view and corresponding baseline [13]. Anomaly detection is the problem of finding patterns in data that do not follow expected behavior, i.e. differ from the baseline. However, the difference between normal and anomalous is often not precise [15]. Hence, there exist numerous methods that analyze that difference and classify it as an anomaly or not.

2.3.1 Anomalies

In [15] the authors describe different anomalies and anomaly detection techniques. They state that there are three types of anomalies:

- Point anomaly - an individual data instance that lies outside the boundaries of normal behavior, i.e. is anomalous with respect to the rest of the data
- Contextual anomaly - a data instance that is anomalous only in a specific context, but not outside that context
- Collective anomaly - a collection of data instances that is anomalous with respect to the rest of the data set; the individual data instances might not be anomalous by themselves

[5] propose a different classification of anomalies. They define network operation anomalies, which are typically power outages. Another type of anomaly is a flash crowd anomaly. A flash crowd is an unusually large demand for a resource/service [32], and could be caused by a software release, for example [5]. The third type of anomalies are network abuse anomalies. These include DDoS and port scans and their purpose is to harm the network, resources or services. According to [46], different types of scanning and DDoS attacks are the most prevalent anomalies.

Distributed Denial of Service

A Distributed Denial of Service (DDoS) attack is an action that aims to make a network resource or service unavailable to its users. This is usually achieved by flooding the destination with multiple service requests such that the server overloads and crashes or all resources are consumed and the service becomes unavailable. The source of these requests can be a single source (DoS) or multiple sources (DDoS). In both cases the attack destination is characterized by a dominant IP address and port number. The attack is volume-based and causes an increase in flow and packet traffic [32]. A DDoS attack in a cloud data center can be harmful by consuming a lot of resources and causing link load balancing problems. DDoS attacks are a contemporary issue as evident from these recent articles: [18], [3], [19].

Port scans

A port scan is an attack that sends requests to a range of server ports looking for an active port, which can be used to exploit a known vulnerability of the service. There are two types of port scans - horizontal and vertical. The vertical port scan attack sends requests to a range of ports on a single server, whereas the horizontal port scan attack sends requests to a specific port on multiple servers. It is characterized by increased flow and packet traffic from a dominant source with similar number of packets and flows [32].

2.3.2 Anomaly Detection Modes and Metrics

According to the same authors, [15], there are also three anomaly detection technique modes, which are classified by how much of the data used is labeled. The supervised anomaly detection uses a training set for the anomaly class and for the normal class. The semisupervised detection labels only the normal class and allows for more flexibility on anomalies. The unsupervised class does not require any training data at all. Another aspect by which anomaly detection techniques differ is the output of the detection. It can be in the form of scores, which show how far from the normal behavior an instance is. Then it can be decided above what score the instance is considered an anomaly. Alternatively, the output can be labels to each test instance - normal or anomaly.

Some metrics for anomaly detection are volume metrics (number of bytes, packets and flows), feature entropy (for example, Shannon entropy) based on flows, IP or port, unique IP counts, and port counts [13, 45]. Volume metrics are suitable for volume based attacks such as Distributed Denial of Service, and entropy based metrics are appropriate for non-volume based anomalies such as portscans, usually associated with worm/virus propagation. Entropy summarizes the distribution of a desired metric such as IP address or port in a single number [35]. It allows for detecting concentration or dispersion of feature distributions that are typical for certain types of attacks [46]. Shannon entropy is defined as $H(X) = \sum_i P(x_i)I(x_i)$, where H is the entropy of a discrete random variable X with possible values $x_1, ..., x_n$, $P(X)$ is a probability mass function, and I is the information content of X. Tsallis entropy is another type of entropy, preferred by [46], because it allows one to focus on different regions of a distribution and, thus, it has advantages over Shannon entropy. The same authors introduce more traffic features, whose distributions can be used as metrics: autonomous system distribution, country code distribution, average packet size per flow distribution, and flow size distribution. Factors that might affect detection of anomalies include a sampling rate, anomaly traffic characteristics, baseline traffic characteristics, and anomaly metric [45].

ASTUTE, [42], (A Short-Timescale Uncorrelated-Traffic Equilibrium) is a different method based on finding correlated flows. The authors claim that it is a good technique for small volume anomalies, and it is best combined with a Kalman filter, which is suitable for large volume-based anomalies [42].

2.3.3 Anomaly Detection Techniques

A number of anomaly detection techniques exists which are presented in [15]. For example, such detection techniques are Bayesian networks based, neural networks based, and support vector machines. The rule-based detection system consists of rules that describe the normal behavior. When an instance does not correspond to the rules, it is considered an anomaly. The basics of all these techniques is that they are trained to learn normal data by using labeled training data, and classify a test instance as normal or anomalous based on the learned model. Hence, they are called classification based anomaly detection techniques. Their advantages are using powerful algorithms to classify instances, and fast testing. The disadvantages of this type of techniques are relying on accurate labels and assigning labels instead of scores. A classification method evaluated by [46] is Support Vector Machine (SVM). This is a learning model, which is trained with a set of data including normal behavior and anomalies, and then validated with another set. The model learns to assign data to a category - normal or

anomalous. The output of this model is a label, corresponding to the category.

Another type of technique is the statistical anomaly detection, an example of this being statistical profiling using histograms. It uses histograms to describe the normal data. The histogram is built based on different values of features of the traffic data. Detection is performed by trying to check if the test instance falls into any one of the bins of the histogram. In statistical anomaly detection techniques normal data instances occur in high probability regions of the model while anomalies appear in the low probability regions [15].

Clustering-based anomaly detection techniques are another type of a detection method. The basic principle is that normal data instances belong to a cluster in the data while anomalies fall outside any cluster. A variation of this principle is normal data defined as instances that belong to large and dense clusters, whereas anomaly instances belong to small or sparse clusters. The advantages of clustering-based methods are that there is no training or labeling of normal data needed, and that testing is fast. However, computational complexity for clustering the data is often a bottleneck [15].

The nearest neighbor-based technique is the next type of anomaly detection. The assumption in this model is that normal data intstances occur in dense neighborhoods and anomalies occur far from their closest neighbors. The anomaly score is based on the distance to k^{th} nearest neighbor or the relative density. This distance can be measured in various ways, the most popular being the Euclidean distance. The advantages of this method are that it does not make any assumptions about distribution, and adapts easily to different data types. The disadvantages include defining distance measures, but mostly the problem is the computational complexity of testing (n^2) [15].

Another type is spectral anomaly detection. The principle behind it is based on data variability. The detection method tries to find an approximation of the data using a combination of attributes, which represent that data variability. Several methods use PCA (Principal Component Analysis) to reduce dimensionality of the data set. In training, principal component values for normal data will be established. In testing, each point will be compared with the principal components and an anomaly score will be assigned based on the distance between the test point and the principal components. The main advantage of spectral techniques is the dimensionality reduction. A disadvantage is that they are only useful if the normal and anomalous instances can be distinguished from each other in the lower dimension [15]. Spectral anomaly detection is also adapted by [5]. They use a variety of tools including simple statistics, time series analysis, and wavelet analysis to extract anomaly features. Wavelets divide data into frequency components enabling analysis of each component separately. [46] evaluates the Kalman filter, PCA, and KLE (Karhunen-Loeve Expansion) as detection methods. The Kalman filter models normal traffic as a measurement-corrected auto-regressive process plus zero-mean Gaussian noise. The difference between the normal traffic model and the evaluation data is a residual, which is calculated for each of the input time series. PCA condenses information to a single output time series. It compares how closely the evaluation input matches the normal behavior model. PCA has a parameter k, which determines how many of the components are used to create the normal activity model. KLE accounts for temporal correlation in the data and is an extension of the PCA method. The only difference is an additional parameter, which indicates how many time bins should be included when accounting for temporal correlations. Another parameter that is common for all of the discussed detection methods is the threshold for the anomaly score.

2.4 Virtual Service Migration

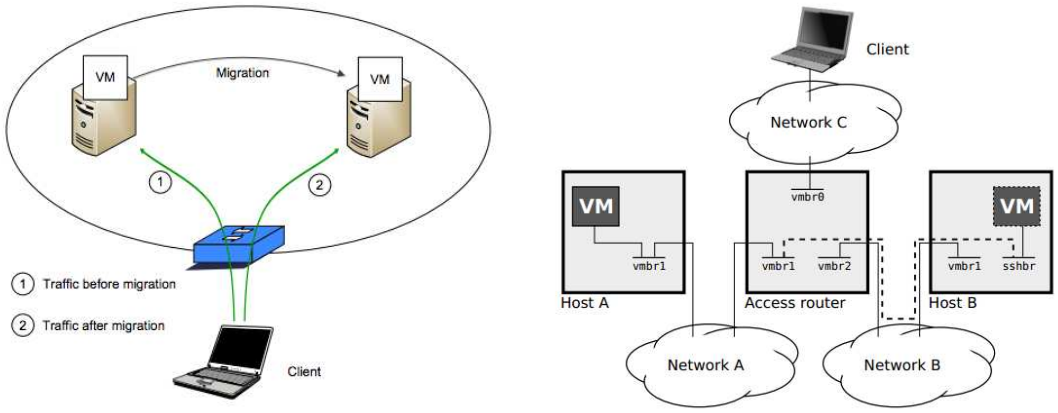
Virtualization reduces dependency on specific hardware through hardware abstraction [22]. Additionally, virtualization of services or resources also allows for dynamically allocating more resources. It does not bound a service to a physical machine, and it can be transferred easily, i.e. it can be migrated. This section gives background information on how and why virtual service migration is done.

2.4.1 Reasons for Migration

Reasons for migrating virtual services can be load balancing for improving resource consumption, maintenance of a server, or failure. In addition, when a service is migrated, related services should also be migrated to ensure that connections are maintained [9]. For example, a web service needs to access data from a database server hosted on the cloud. When the web service is migrated, the database server should also be moved to the new location to maintain a fast connection with the web service. A reason for a migration across different subnets can be a large-scale failure due to a natural disaster or a power failure [22]. The purpose for migration can also be to reduce network costs and move services closer to the users, in a different physical location [10].

2.4.2 The Migration Process

There are two different scenarios, in which migration can be performed. Cold migration occurs when the virtual machine is stopped and then transferred. It is less complex, but it also requires a longer service downtime. In contrast, in a live migration the virtual machine is running while the device state is transferred to the new location. However, if the reason for migration was failure, it is not possible to perform a live migration because the VM is not running [9]. A virtual machine migration consists of transferring the file system, the CPU state, the RAM contents, and redirecting network traffic to the new location of the VM [22]. The virtual machine continues to run while memory is being transferred to the destination and is only terminated when everything possible has been copied to the new location [39]. Once it is suspended, the traffic is redirected to the destination host, and the migrated VM is activated [14]. Snapshots of the current state can already be replicated in different locations before the migration has started. In that case the file system is already at the destination and does not need to be transferred [22]. It is a good idea to balance the number of replicas of each virtual server and before migration decide whether a new replication should be done or an existing one should be used as the destination. This is important, because the server state is typically large and the traffic volume to be generated by migration should not be neglected [10]. Virtual machines have their own MAC address so they can be moved while they are running and still keep their network connections as long as the new machine is in the same subnet as the original [39]. The application downtime caused by migration is empirically shown to be less than a second [39].



(a) Redirecting Network Traffic (b) Wide-area migration enabled by tunneling

Figure 2.2: Virtual service migration [22]

2.4.3 Types of Migration

There are two broad types of migration - local area migration and wide area migration. Wide area migration moves a virtual machine across different subnets compared to migrating the VM

to a different server in the same local subnet. It also requires changing the IP address, according to [11]. In addition, the routing of network traffic has to be adapted and network traffic redirection, illustrated in figure 2.2(a), has to be performed because the location of the service is now in a different subnet [22]. There are different ways to redirect traffic for a wide area migration. [22] suggests several approaches, which we will now discuss in some detail.

Approaches using an additional device

One approach is to use an application layer proxy at the original network to forward application layer traffic to the VM after successful migration [22]. Another is implemented at layer 3 and makes use of the Mobile IP protocol [22]. Traffic sent to the original VM is first received at a Home Agent, which knows of the migration and forwards the traffic to the new location. This idea is also supported by [43]. Yet another approach is to hide the VM behind a Network Address Translation router [22]. Thus, when a request is sent to the external address of the NAT router, it will be forwarded to the new VM. Another suggestion is to use a VPN gateway to which the VM connects after successful migration [22]. A drawback of these solutions is the indirection point. It can take time to update it with the new location, but this can be controlled by setting a short Time To Live for the cache. A similar issue with this approach of traffic routing is the delay added by sending requests through an additional agent. Another reason the indirection point can be a problem is that it is not known how long it will be available. If the migration was initiated due to power failure or natural disaster, that indirection point can also be affected.

Tunneling approach

If the access router of a service is not affected by a failure, it can be used to implement a different approach called Link Tunneling [22]. A virtual link is created between the original access router of a service and its new location, and it is used to redirect any incoming traffic to the new location. Figure 2.2(b) shows the network setup used for that migration. The downtime due to migration for this approach is 0.6s according to experiments conducted in [22]. [43] says that downtime for wide-area migration is the same as the downtime for local subnet migration. In the above approach, the virtual machine does not need to change its IP address. Another proposal, given by [11], focuses on IP tunneling with Dynamic DNS, which helps overcome breaking connections when the service changes its IP address. The tunnel is torn down when there are no more connections that use the old IP address. This also allows the VM to continue running through migration.

SIP signalling

A solution involving SIP signalling (Session Initiation Protocol) requires an IP address change, but on the other hand it does not rely on additional devices. In this approach, user accounts are subscribed to events about connectivity. When a service is migrated and there is a change in its IP address, the users receive a notification about the change and can now send requests to the new location [22].

Table 2.1 summarizes different reasons for migration and gives information in what situation which type and mode of migration can be used and which migration approaches are applicable.

2.5 Conclusion

Several topics have been discussed in this section. Starting with network data centers architecture, typical infrastructures consist of two or three levels composed of ToR, Aggregation and Core switches, which can store routing tables for the rest of the switches in the data center. Addressing between these switches and between servers is done separately and mapping of addresses is used to determine flow paths. Traffic flowing between these devices is internal traffic, which represents 80% of overall data center traffic, includes a vast range of services and is complicated to model. Traffic is usually analyzed at the edge of the data center at the core

Reasons for migration	Migration type	Migration mode	Approaches available
Load balancing	local/wide	cold/live	all
Maintenance	local/wide	cold/live	all
Reducing network costs	wide	cold/live	all
Local failure	local/wide	cold or live (only if surrounding devices have failed, but not the server hosting the VM)	if the indirection device has failed, use SIP signalling
Natural disaster/power failure (large-scale)	wide	cold	approaches without indirection points (use existing replication and redirect traffic)

Table 2.1: Reasons for migration and some properties

switches after reducing its amount, but there can be various consequences of sampling traffic. In addition, traffic is analyzed for the purpose of anomaly detection. The most popular anomalies to detect are DDoS and port scans. They are detected based on features collected from the traffic, and there are different types of techniques for analyzing traffic and detecting anomalies. Finally, virtual service migration is introduced to the data center structure and traffic. There are two types of migration - local and wide-area migration, and they affect data center traffic differently. Hence, they could also affect anomaly detection. This work aims at showing if and how virtual service migration affects the results of anomaly detection of traffic data in clouds.

Chapter 3

Methodology

This chapter starts by describing the conceptual system of the cloud data center, including infrastructure, cloud traffic, service migration and anomaly detection. Based on recent research, it summarizes the typical cloud data center, which is necessary to get as close as possible to a realistic overview of how migration affects correctness of traffic analysis from existing anomaly detection techniques. Once this is achieved, the chapter continues by describing the experiment methodology, which includes every step from what data is used in the experiments and how it is modified to quantifying the affect on anomaly detection precision after migration. Additionally, these steps involve existing tools and frameworks that are described at the end of the chapter.

3.1 A Cloud System

As discussed in Section 2.1, the overview of a typical cloud system includes its data center architecture and internal and external traffic. Summarizing the infrastructure is necessary in order to determine where in the architecture traffic is analyzed for the purpose of anomaly detection. Moreover, real cloud traffic data or data that can simulate cloud traffic is needed for these analyses. In addition, virtual service migration, which is the focus of this work, and its effect on traffic need to be simulated.

3.1.1 Data Center Architecture

This is a description of a typical data center architecture, based on research presented in the background chapter. The proposed system is a 3-tier data center architecture. It consists of core switches at the top level. Next, there are aggregation switches, which are connected to the core switches via a Clos network, such that every aggregation switch is connected to every core switch. This allows for a larger choice of flow paths. The third tier of the architecture consists of the ToR (Top of Rack) switches. Each of the ToR switches is connected to two aggregation switches for redundancy. It is also connected to a rack of servers. Each server can host multiple virtual machines depending on its resources and on the demands of the services. Figure 3.1 illustrates this 3-tier architecture. The advantage of this architecture is horizontal scalability. Additional switches can be added horizontally to the network, expanding without disrupting the existing architecture.

The allocation of addresses in this data center architecture, similarly to [26, 38], is achieved by using different layers for service addresses and switch addresses. All the services are assigned internal private addresses. Thus, the switches can obtain the complete switch level topology, and the services see all other services as if they are on the same private network. ToR switches store the private addresses of the services they are connected to. A mapping for each server's private address to its ToR public address exists in a directory system, hosted on several servers in the data center. When one server wants to communicate with another, the packet sent is encapsulated at the corresponding ToR with its public address and sent to the destination ToR's public address. It is decapsulated when it arrives at the correct destination ToR. The only exception stands for services that need to be directly reachable from the Internet,

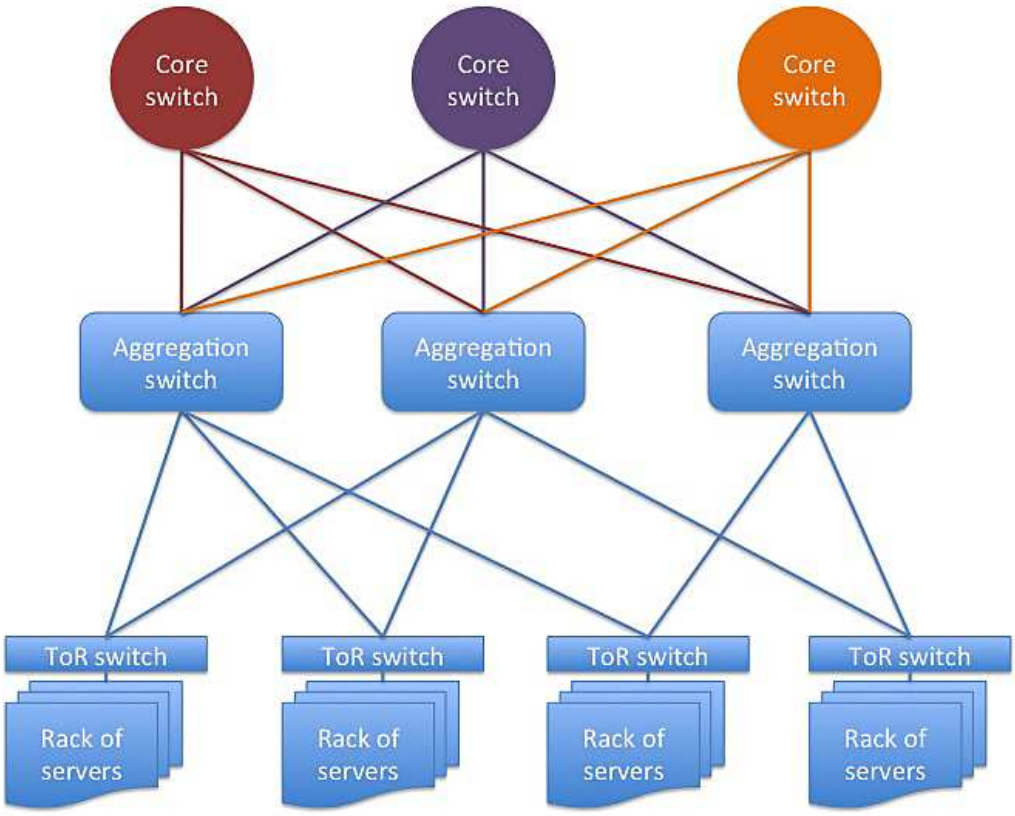


Figure 3.1: A typical cloud data center infrastructure

such as web servers. They are assigned their own public address in addition to the private one. The purpose of the Clos network between the aggregation switches and the core switches is to have a fault-tolerant architecture, and also more options for load balancing. With a lot of paths available between two services, a flow can choose a random path to its destination. However, in this data center architecture, the path of a flow would be hash-based. Thus, in the traffic simulations that are presented later in this work, when an experiment is repeated, the same flow will take the same path, but different flows will take different paths. Consequently, if a change occurs, the reason will not be an unpredicted flow path, but an anomaly or migration.

As mentioned earlier, services are assigned private addresses. This address is not supposed to change, which means that when a service is migrated to another server in the same data center, which is connected to a different ToR switch, the service will keep its original address. The only thing that will change is the mapping of addresses in the directory system. The directory system will be updated right away, but the ToR switches caches will not. They will only be updated when they receive a packet for a service they do not host and ask for the latest address mapping. This will avoid additional amount of traffic after a local area migration. However, during wide-area migration, depending on the method used, the service address might change.

3.1.2 SWITCH Traffic Data

SWITCH traffic provides us with real data, which we use to analyze real traffic towards existing clouds and also analyze existing services hosted on the network. Traffic data is a significant part of the cloud system. Since traffic generated from scratch will not be realistic, it is best to use already existing traffic data. This work is done in cooperation with SWITCH [44], a medium-sized backbone operator that connects several Swiss universities, research labs and governmental institutions. All these institutions are part of the SWITCH internal address range. SWITCH stores all network traffic data that passes through their border routers without applying any sampling, hence, it only stores incoming and outgoing traffic. The set of data used covers a week of data between October 30, 2012 and November 5, 2012. This traffic data has to be analyzed and

compared to real cloud traffic data before it can be used to simulate cloud traffic.

SWITCH NetFlows

SWITCH traffic data is stored using the NetFlow standard. According to Cisco [17], a NetFlow contains the source and destination IP address, the source and destination port, and the IP protocol for a unidirectional sequence of packets. A NetFlow packet header provides basic information about the packet such as the NetFlow version (in the case of the data used here, this is version 9), a sequence number, used to detect lost packets, and a number of records within the packet. A NetFlow record also includes a lot more information about the flow, but what is relevant for this work is the following: timestamps for the flow start and finish time in Unix seconds, number of packets in the flow, number of bytes in the flow, the id of the export device, source and destination IP addresses, source and destination IP ports, and the IP protocol.

Due to previous research using this week of data, a processed version of it is available for analysis. Let us define a service as a unique pair of an IP address and a port number. Then this processed data includes flow count information for each service recorded by the SWITCH border routers for every 5 minutes in the selected week. It also includes a summary file with a total count of flows, packets, and bytes per service, which holds this information for 3,253,723 external and internal services. The 5-minute-files also contain flow direction information (0 - incoming, 1 - outgoing), which can be used to separate incoming and outgoing services in order to draw individual conclusions for both. This distinction was used to calculate a total flow count for each direction for each 5-minute-file. A 5-minute-file contains information about the total number of flows for each service. To calculate the total flow count, sum all the flow counts of services that match the desired traffic direction. The results can show a pattern of traffic behavior. Figure 3.2 shows the flow count for all outgoing SWITCH traffic per 5 minutes. This includes traffic data from requests to externally hosted services issued from an instituion within the SWITCH network or replies to requests for a service hosted within the SWITCH network. The pattern does not appear to be bursty, and most of the flow counts in duration of 5 minutes range between $0 * 10^6$ and $4 * 10^6$ with higher density around $1 * 10^6$.

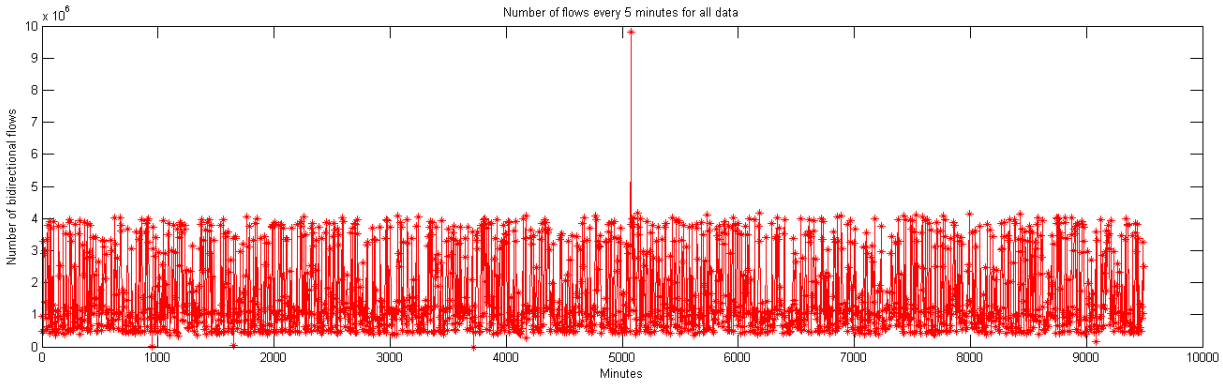


Figure 3.2: Flows per minute for all outgoing SWITCH traffic

Figure 3.3 also shows flow counts per 5 minutes, but for incoming traffic. These can be requests for services hosted on the SWITCH network or replies from services outside the SWITCH network for requests issued within the network. The flow counts range mostly between $1 * 10^6$ and $2.2 * 10^6$. Traffic is again not bursty, with a couple of exceptions.

Amazon cloud data

Since the goal of this work is to determine how virtual service migration affects anomaly detection in clouds, it would be ideal if actual cloud data can be used to simulate the cloud traffic. One of the biggest cloud providers is Amazon. It hosts services in cloud data centers all around the globe. Since the traffic data available is from a European network (SWITCH in Switzerland), it would be best to just consider services hosted on Amazon Europe, since it is more likely

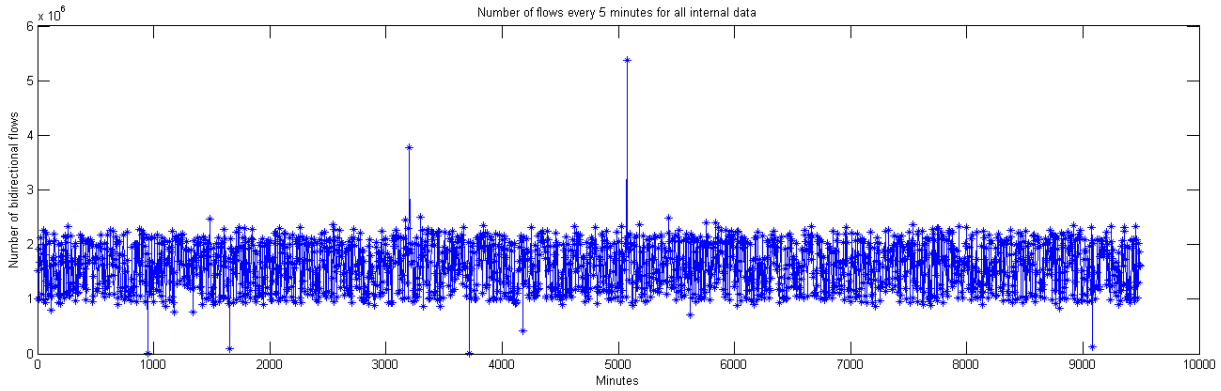


Figure 3.3: Flows per minute for all incoming SWITCH traffic

services hosted in Europe will be accessed from a European network. According to the official Amazon forum, [2], the IP ranges for the European cloud data center in Ireland are the following:

79.125.0.0/17 (79.125.0.0 -79.125.127.255)
46.51.128.0/18 (46.51.128.0 - 46.51.191.255)
46.51.192.0/20 (46.51.192.0 - 46.51.207.255)
46.137.0.0/17 (46.137.0.0 - 46.137.127.255)
46.137.128.0/18 (46.137.128.0 - 46.137.191.255)
176.34.128.0/17 (176.34.128.0 - 176.34.255.255)
176.34.64.0/18 (176.34.64.0 - 176.34.127.255)
54.247.0.0/16 (54.247.0.0 - 54.247.255.255)
54.246.0.0/16 (54.246.0.0 - 54.246.255.255)
54.228.0.0/16 (54.228.0.0 - 54.228.255.255)
54.216.0.0/15 (54.216.0.0 - 54.217.255.255)
54.229.0.0/16 (54.229.0.0 - 54.229.255.255)

Earlier it was mentioned that there exists a preprocessed summary file for all the services passing through SWITCH border routers for the selected week. In order to just observe services hosted on the Amazon cloud, the list of all services needs to be filtered by the above Amazon IP ranges. It is important to mention that at the time of processing this data the last three ranges in the Amazon list were not published, hence they were not included in the filtering. However, since they were recently introduced, in any case there should not have been any services with IP addresses in those ranges. The Amazon hosted services were obtained by going through each service in the summary file and checking if the IP address is contained in one of the IP ranges. When it was contained, the available information about the service was stored. There are 6,275 unique services hosted on the Amazon range that are accessed through the SWITCH network. The purpose of obtaining all the Amazon hosted IPs is to get a list of IPs, and then for each IP get flow count data from the more detailed 5-minute-files. Just as flows for incoming and outgoing traffic were counted separately by filtering by flow direction, flow counts for every 5 minutes were calculated by filtering by the obtained Amazon IP list. Figure 3.4 shows the total flow count for all outgoing SWITCH traffic to Amazon hosted services for every 5 minutes. The pattern does not appear to be bursty, and most of the flow counts in duration of 5 minutes range between $0 * 10^4$ and $3 * 10^4$ with higher density around $0.2 * 10^4$. If it is compared to Figure 3.2, it can be noticed that both sets of traffic follow a very similar pattern. Hence, the Amazon subset of data is a good representation of the overall traffic. The opposite conclusion can also be made: the SWITCH traffic looks a lot like traffic to a cloud data center.

The analysis of the Amazon subset of traffic data also yielded additional results. Since a service is defined as a unique IP address and port pair, once the IPs in range are extracted, the ports used by those services can be observed. The IP protocol of the ranged services was either TCP or UDP, with HTTP and HTTPS services accounting for most of the TCP services, and UDP based DNS services giving the highest flow counts. Hence, it can be concluded that web

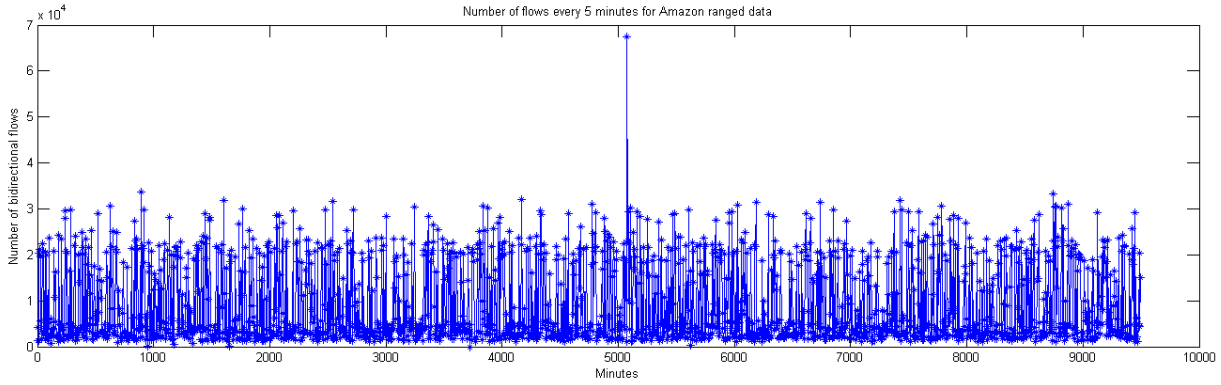


Figure 3.4: Flows per minute to Amazon hosted services

services are the most popular services hosted on the cloud data center.

External traffic

The goal of analyzing the SWITCH traffic and the Amazon cloud traffic is to extract suitable services to simulate traffic in the designed cloud infrastructure. One possibility is to use data from real cloud services such as the obtained traffic from Amazon hosted services. However, data traffic for those services is only available for requests sent from the SWITCH network. Hence, the traffic for a particular cloud service is incomplete and the data is just a small subset of the complete service data. A different possibility is to use SWITCH hosted services as the simulated cloud services since the complete data traffic for those services is available. It has already been shown that the Amazon subset traffic and the SWITCH traffic follow the same flow distribution pattern. Thus, it can be claimed that using SWITCH services instead of Amazon services will give the same results. In addition, a cloud can host any kind of service, including the ones that are hosted on the SWITCH network, thus making the SWITCH services potential cloud hosted services. Moreover, later it is shown that the used services are also mostly HTTP and HTTPS web services just as the available Amazon services.

Based on existing research presented in the Background Chapter, internal data center traffic covers bulk, network files, backup, data mining, index computations, MapReduce, as well as internal requests between services. Moreover, internal traffic's flow patterns vary depending on the assignment of virtual machines between physical servers. It accounts for 80% of overall data center traffic. All this information shows the complexity of internal traffic and its behavior within the data center. Thus, modeling internal behavior can be very complicated especially when there is no internal data available to use. Hence, this project focuses on analyzing external traffic only, focusing on attacks such as DDoS and port scans that originate from outside the data center and are visible in the external traffic.

3.1.3 Migration

Another aspect of traffic behavior is virtual service migration. Disregarding the type of migration, expected traffic behavior will change after a service has been moved to a different physical location. In the case of a local-area migration (for example from one rack of servers to another rack of servers due to resource management between the servers), the traffic generated by the migrated service will follow a different route and pass through different switches. Thus, ToR and Aggregation switches will notice a change in traffic, whereas core switches might not, depending on the new location and flow paths. When a wide-area migration occurs, traffic pattern changes are more visible. Since a virtual service is moved to a different data center (for example for the purpose of reducing network costs), the traffic for that service will completely disappear from the original data center once the last steps of the migration are completed. From the view point of the destination data center, traffic for a new service that was not hosted there earlier, will appear. Migration will affect traffic patterns in another way, too. According to

the migration process, before traffic can be redirected to the new service location, the state of the service needs to be transferred. This transfer will generate additional traffic. In a local-area migration this will be part of the internal traffic; in fact, it is a replication of a service. In wide-area migration, the external traffic will be affected because the transfer will be done between different data centers. The affect on the traffic change by the virtual machine state transfer largely depends on the amount transferred. It could be the case that a completely new virtual machine is allocated for the migrating service. Then, the whole system state and memory have to be copied to the new location. However, it could also occur that the migration is directed to an already existing replication of the service. In that situation, only the most recent memory state has to be transferred and it will have a much lower affect on the expected behavior. In any case, replication of services of various sizes is done regularly in data centers for redundancy or load balancing, and is thus considered part of normal traffic behavior.

In the scenarios presented later in this work, the focus is on wide-area migration. One of the reasons is the complexity of modeling internal traffic. Without internal traffic, local-area migration cannot be observed or simulated because it only involves internal traffic. Moreover, local-area migration does not affect traffic as much as wide-area migration because the latter causes all traffic towards a service to stop appearing in the current data center, whereas in local-area migration the traffic is just taking different paths to its destination. Another reason is the method of anomaly detection, which is discussed in the next subsection of this chapter. In the following experiments, migration will be modeled based on the traffic changes it creates. One of them is the additional traffic for replicating the service when the new VM location does not have an existing replication of the service state. However, since service replication is considered expected traffic behavior, it is not necessary to model it additionally. The second change is the one that occurs after migration is completed - redirection of traffic. From the data center's viewpoint traffic towards a specific service, which has been migrated, will stop. In the destination data center, traffic for a new service will appear. Hence, modeling migration in the original data center will start with the service normal behavior and at a certain time the requests towards that service will cease.

3.1.4 Anomaly Detection

The last step in simulating a cloud system is determining the type of anomaly detection technique used, what traffic it analyzes and where in the infrastructure that traffic is collected. According to research presented in the Background Chapter, traffic analysis of incoming and outgoing traffic is done at the entry point of the data center. In the presented typical cloud data center infrastructure this entry point is the layer of core switches. Hence the core switches will be the point in the architecture where traffic will be collected and then analyzed. Traffic at each core switch can be analyzed individually or together. If it is analyzed by core switch, then there is less traffic data to be processed. However, some anomalies do not target a single service such as a DDoS attack, but are spread among many services such as a horizontal port scan, where the target is a specific port number in a group of IP addresses. The services to which those IPs belong are likely to be spread throughout the datacenter and not be covered by the same core switch. Hence, when traffic is analyzed from a single core switch only, then only partial data of the attack will be available, and this data might not be enough to detect the anomaly successfully. Thus, it is better when traffic data from all core switches is analyzed together. Individual service attacks will still be detected, but so will attacks on multiple services. This idea is similar to the traffic mix proposal in [45].

The most popular anomalies that need to be detected by this cloud system are DDoS and port scans. DDoS is a volume-based attack against a single destination IP. Its features include spikes in flow and packet count. Thus the anomaly detection technique chosen needs to analyze these traffic features. Port scans also show a spike in flow traffic, but are nonvolume-based attacks. Their attack can target a specific port on multiple IPs or multiple port numbers on a specific IP and is, thus, better detected with entropy-based features such as an entropy based on IP and port distribution. Moreover, according to existing research [13], entropy is more robust to traffic processing. Hence, the anomaly detection technique chosen for this cloud system will need to analyze both volume based features such as flow, packet, and byte counts, and entropy based

features such as IP and port distribution.

Another important point in traffic analysis is that only external traffic will be analyzed, as mentioned earlier. Even though local-area migrations are not considered in this work, with the current system configuration they will not be detected. If a local migration causes traffic to pass through a different core switch, this traffic will still be part of the overall traffic data just as before the migration because when traffic is analyzed all of it is gathered together. Thus, hypothetically, local-area migration should not be detected as an anomaly if internal traffic was also analyzed at the core switches level. However, this will not be pursued further in this paper.

3.2 Experiments

So far it has been decided how every part of the cloud system will be simulated. This section shows how the decisions made are incorporated using scripts and existing tools to produce final results. The goal of the experiments is to show if virtual service migration affects results from anomaly detection and if yes, how much, and what these results depend on. Figure 3.5 shows all the steps in conducting an experiment from getting SWITCH data to the percentage of false positives with and without migration. The experiments methodology starts with processing raw NetFlows and storing traffic data for selected services. Additionally, different anomalies are injected and stored in the same format. Then experiment details are decided such as subset of traffic data analyzed, anomalies injected, and services migrated. The traffic data is used to create a timeseries file consisting of traffic features, which is used as an input for an anomaly detection tool to evaluate anomalous traffic. Based on the cloud system description, anomaly detection is done on all of the available traffic data together. Hence, during the experiments it is not necessary to determine the exact flow paths through the architecture because the anomaly detection does not need this information. It only requires the features of the traffic data for a set of services.

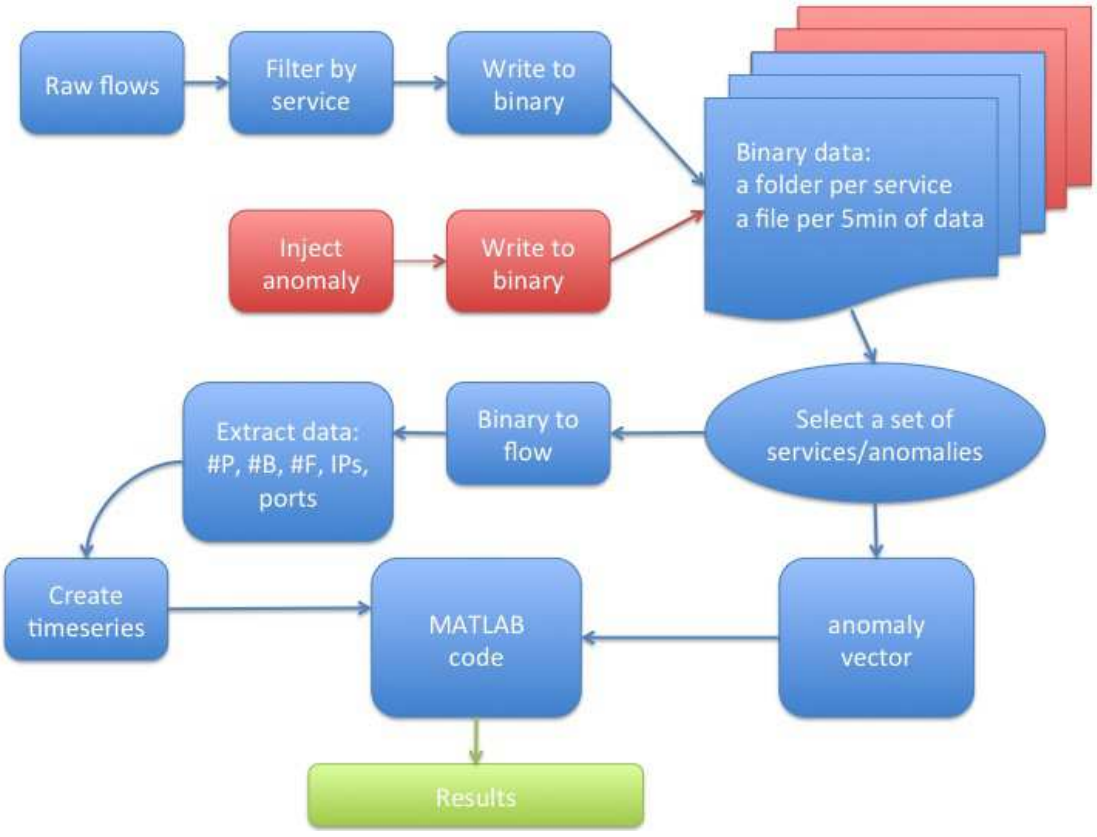


Figure 3.5: Methodology steps

3.2.1 Cloud Services

Before any experiments can be run, traffic data has to be processed. As decided earlier, this traffic data will be actual SWITCH services, simulating cloud services. A cloud can host a large amount of services, but it will not be practical to conduct experiments with real-size cloud traffic because the amount to process will be too large and because we do not have access to real-size cloud traffic. Hence, it has been decided that experiments will be run and conclusions will be made based on a subset of services. Since it is desirable that the used services have as much data as possible, the top 200 services hosted on SWITCH have been selected (Raw flows in figure 3.5). The top services were chosen based on their total weekly flow count. They were extracted from the summary file. First, a list of all internal IP addresses was extracted by using the location parameter in the 5-min files. Then, based on this list, all data for internal services was extracted from the summary file. Once total counts for all the internal services were available, the top 200 out of 291,582 services were selected based on highest flow counts for the week. These top 200 services include both TCP and UDP services, and just as the observed ones hosted on the Amazon cloud, they are mostly web services (HTTP and HTTPS) and DNS services but also include a couple of mail servers and some private services.

SWITCH data is stored in a compressed raw NetFlow format. FlowBox [24] is a framework for stream-processing of voluminous flow data and can be used to read the SWITCH NetFlows. The toolbox provides a flow structure, which is how each flow is stored when read. This framework was extended for the purpose of this work. The raw data contains flows for all SWITCH services and we only need the flow data for the top 200 services. Hence, the framework extension, called *flow_to_bin_cc*, was implemented to compare the source and destination IP address and port of each flow to the list of top 200 services (Filter by service in figure 3.5). A binary file for each of those 200 services was created to store any flow that was sent to or from the service. Flows were stored in binary in the order they were read. It is important to note that this order is not necessarily sequential. Shorter flows can be processed faster and thus arrive sooner at the reader. Longer lasting flows might need to be cut if they last more than 5 minutes, and will thus arrive later at the reader than a flow that started after them. However, it can be expected that at any selected time a flow older than 20 minutes will not be read. Each flow contains information necessary for extracting features for the timeseries such as packet and byte counts, source and destination IP address, source and destination port number, and flow starting time in UNIX seconds. The latter is needed for processing the binary files further.

Once all raw flows are read, the FlowBox framework needs to be extended further to continue processing the 200 binary flow files. Since the data granularity of the flows is coarse, it does not need to be reduced in size, but handled in such a way that the timestamps' lack of precision will not affect the traffic analysis. Flows can be grouped together in 5 minute bins (Binary data in figure 3.5). Thus, when timeseries are created, the traffic features will be extracted on per 5-minute-base. To achieve this, the flow start time can simply be divided by 300s. The result is an index, used to denote the file to which the particular flow needs to be written. In a few words, the second extension of the flow reading framework, called *write_to_5min_bins*, reads each of the 200 binary files, and writes each flow to a particular file depending on its starting time. Earlier, 5-minute files were mentioned. However, they cannot be used because the data contained in them only includes total flow counts, and is missing information necessary for the timeseries extraction. Writing the new 5 minute files seems straightforward, however, it becomes a little complicated because for each service there are more than 4000 files to be written. They cannot be open all at once because the system's limit allows up to 500 open files. The solution comes with the fact that flows might not be ordered, but they are not completely scattered either. At any selected time a flow older than 20 minutes will not be read. Thus, the extension requires only 5 open files at a time. They are handled using pointers to the file descriptors. When a flow is read, for whose index the correct file is not open yet, the oldest file is closed and its pointer is assigned the address of the new file descriptor. Once all flows are split in 5-minute-files, they can be processed by file index to create timeseries for the anomaly detection tool. These timeseries will be the baseline for normal traffic.

3.2.2 Anomalies

Anomalous traffic is baseline traffic plus injected anomalies. In these experiments, there will be two types of anomalies injected into normal traffic - volume (DDoS) and non-volume (port scans) based. The goal is to first observe how well the detection tool identifies anomalies, and then to compare to how well it identifies anomalies when there is migration occuring in the same trace. Injecting the anomalies into the normal traffic is done using the Flame tool [23] (Inject anomaly in figure 3.5). Flame requires a model for the anomaly and a flow, on which to inject it. We use an empty flow for the anomaly injection so that it does not affect the traffic subset analyzed. In the model file the characteristics of the anomaly are specified. This includes source and destination IP, which could be a specific IP such as in the case of the target of a DDoS attack, a randomly chosen single IP such as the source of a port scan attack, or a distribution of IPs such as the source IPs in a DDoS attack. It also includes source and destination port, which can also be specified, randomly chosen or a distribution can be provided. In addition, the probability that the destination service will reply to the attack, the protocol, the number of bytes per packet, and the number of packets per flow are provided. Then an anomaly generator script injects anomalies based on the provided model and specified anomaly intensity. An anomaly intensity of 0.0001 corresponds to 500 injected anomalous flows for the 5 minute range. The anomaly intensity has to be determined with respect to the amount of traffic in the baseline. This has to be figured out empirically by testing the selected baseline data with different anomaly intesities. The strength of the anomaly should not be too weak because then not all of the anomalous flows will be identified correctly, resulting in false negatives, or bursty traffic part of the baseline could be classified as an anomaly (false positives). False positives results are baseline traffic which was incorrectly classified as an anomaly. False negative is an anomaly, which is incorrectly classified as normal traffic. On the other hand, if the anomaly intensity is too strong, then a small change in traffic behavior such as a migration or a different smaller anomaly will not be detected.

When anomalies are generated using Flame, they are stored in raw flow format, just as the original SWITCH data. The FlowBox framework is once again used to read flows (Write to binary in figure 3.5). However, this time they do not have to be manipulated in any way. Since the generated anomaly is for a range of 5 minutes, when the flows are read, they only need to be written in binary format to a single file. This file can then be duplicated for other timestamps. All the anomaly files are stored with the correct file index (timestamp divided by 300s), so they can later be read by the script creating the timeseries (Binary data in figure 3.5).

3.2.3 Timeseries

When all the baseline and anomalous data is processed and stored in 5-minute-files, it can be used for the next methodology step - extracting traffic features to create timeseries. This is the *extract_timeseries* extension of FlowBox (Create timeseries in figure 3.5). The timeseries file has an entry for every 5 minutes of traffic data. The first part of the timeseries creation is providing the set of services that need to be included (Select a set of services/anomalies in figure 3.5). Next, for each file index (flow timestamp divided by 300s), i.e. for every 5 minutes, traffic features based on all flows for those 5 minutes are extracted. These features for each direction of traffic are (Extract data in figure 3.5):

- Flow count - number of flows in one direction for all services; count increases by 1 for each processed flow
- Packet count - number of packets in one direction for all services; count increases with the current flow's packet count for each processed flow
- Byte count - number of bytes in one direction for all services; count increases with the current flow's byte count for each processed flow
- Source IP entropy - Shannon entropy based on IP distribution of source addresses; for each processed flow if the source IP is already added to the IP map, increase count by 1, and if not, add it to the IP map with count 1; the IP map contains the IP distribution
- Destination IP entropy - Shannon entropy based on IP distribution of destination addresses; data gathered as for source IP entropy

- Source port entropy - Shannon entropy based on port distribution of source ports
- Destination port entropy - Shannon entropy based on port distribution of destination ports

An entry including all these features is added to the timeseries file for every file index. The *extract_timeseries* extension loops through every file index, then for every service opens the file with that index and reads all flows. The data from each flow is added to the corresponding features. Data for both incoming and outgoing traffic is calculated simultaneously because flows of both directions are mixed in the binary files. Also, entropy is only calculated after all flows for the current file index have been processed and the IP and port distributions have been gathered. Looping through every index and every service and extracting traffic features gives the baseline for normal traffic. The same script can be used to generate anomalous traffic on top of the normal traffic by specifying a folder that contains the anomalous binary files. The same procedure follows, but after the flows for every service have been processed, if there exists an anomaly file in the specified folder, it is also processed exactly the same way as a service file. All flows are processed one by one, traffic features are extracted and added to the total features for that index. Only then they are written to the timeseries file.

3.2.4 Migration

Virtual service migration is processed similarly to services and anomalies, but it is not stored in binary format separately, because a migrating service is an already existing service. Hence, the original service binary files can be used. As discussed earlier, migration is modeled by removing or adding service traffic. When the timeseries files are created, if the migration boolean has been set to true, a service will be migrated (Create timeseries in figure 3.5). This service is chosen from the list of all services before looping each index and reading flows. The service or services selected are saved in a list of migrated services, and then they are removed from the list of baseline services. This is necessary so that the migrated service's flows are not processed twice - as a baseline service and as a migrated service. The baseline of normal traffic is generated with the migrated service as a normal service. A timestamp for migration can be chosen in advance, and during looping through each index the chosen service to be migrated is being processed just like a normal service or an anomaly and it contributes to the traffic features until the migration index has been reached. After that index, the migrated service's flows are not read any more and are not included in the timeseries traffic features. Timeseries that include migration also always include anomalies because the rate at which migration affects false positives and false negatives anomaly classifications depends on the anomaly type and intensity. The effect of migration also depends on the size of the service or group of services migrated. Naturally, migrating a smaller service will have less effect on anomaly detection than migrating a top service. Hence, experiments with migration have to define the percentage of traffic the migrated service accounts for from the total traffic of selected services. Thus, conclusions can me made by finding the threshold percentage of migrated traffic between acceptable and not acceptable anomaly detection results.

3.2.5 Anomaly Detection

So far it has been described how to create a baseline timeseries file, an anomalous timeseries file, and a timeseries file involving migration. The tool that is used for anomaly detection in the next step is called TES tool [47] and it belongs to the spectral anomaly detection type (Matlab code in figure 3.5). It requires the collected traffic features: timestamp, flow, packet, and byte count, and IP and port entropies. For input it uses database tables and an anomaly vector. The timeseries files can be converted to the required databases using a conversion script, which is part of the tool. This script accepts as input the folder where one or more timeseries files are located (for incoming and outgoing traffic, and for different protocols - TCP or UDP), the name of the database to be created and a third file, which specifies the structure of the timeseries files. The databases are named *trainingdata.db* for the baseline traffic and *evaldata.db* for the anomalous traffic or the anomalous plus migration traffic. The other requirement for the TES tool is providing an anomaly vector (anomaly vector in figure 3.5). The anomaly vector is a list of timestamps and id for each timestamp. The id specifies where there is an anomaly and which one. The ids are defined in the beginning of the anomaly vector together with the model of the

injected anomaly and its intensity. This file is necessary so that the tool can compare the results about anomalies it discovers to the correct distribution of anomalies. If they all match, the results will be perfect. For every mismatch, the false positives ratio will increase and the true positive percentage will decrease. The TES tool uses Kalman filter or Karhunen-Loeve expansion with PCA (Principal Component Analysis) to obtain anomaly detection results. It uses the week long baseline data to train its detection system and using PCA with different dimensions it produces results for different anomaly score values. All these results are plotted as a ROC curve (Receiver Operating Characteristic) for each dimensionality. A dimensionality represents how many principal components are used to define the baseline traffic. The ROC curve is a plot of true positives fraction out of all the positives vs the false positives fraction out of the negatives for different thresholds. The goal is to keep the true positives rate (TPR) high and the false positives rate (FPR) low by finding a threshold that gives an acceptable balance between the two ratios. In this project, the desired value for FPR is 2%. This will be the relative ratio to which all the results will be compared and evaluated. Another decision that needs to be made concerns the anomaly score threshold - what score is classified as an anomaly and below what score it is still considered as normal traffic.

3.2.6 Conclusion

Following Figure 3.5, the steps taken to prepare and execute experiments can be summarized. It all starts with the services traffic. The SWITCH traffic data is in raw flow format. Then these raw flows are filtered such that only traffic for the top 200 services remains. Then the flows that compose this traffic are written to binary files, such that for every service there is a folder of files and each file stores flows for a range of 5 minutes. Such files are also created for anomalies. First they are injected using the Flame tool. Then they are written to binary and stored in 5-minute-files. From all this data a subset of services and anomalies is selected to be analyzed. An anomaly vector describes where the anomalies are injected. All the services and anomalies are read from the binary format back to the original flow format. Then traffic features (flow, packet, and byte count and IP and port entropy) are extracted to create a timeseries file. The timeseries can be a baseline or anomalous, and it can also contain a migration of one or more of the included services. The timeseries file together with the anomaly vector are used as an input to the anomaly detection tool, which in turn plots results in the form of a ROC curve.

3.3 Toolset

The processing and analysis of traffic data was done with the help of a few already existing tools.

3.3.1 FlowBox

FlowBox [24] is a ToolBox for online stream-processing of voluminous flow data. It is free software under the GNU Lesser General Public License. The framework is written in C++ and Ruby. The tool was used in this work for processing raw NetFlow data. It was extended to filter traffic by IP address and port number, to store the processed flows in binary format, to read them back to flow format, to split flows based on timestamps, and to extract traffic features from them. The extension was written in C++.

3.3.2 FLAME

Flame [23] (Flow-Level Anomaly Modeling Engine) is a tool for evaluating anomaly detection systems operating at the flow level written in C++ and Python. It uses a stream of flows in NetFlow format, injects anomalies according to a specific anomaly model, and outputs the modified trace. This output can be used with an anomaly detection system to determine if it succeeds or fails to detect the injected anomalies. Each flow attribute is represented by a stochastic model. In this project, the tool is modified to use an empty flow as input, and thus the output trace only contains the injected anomalies. The anomaly models describe DDoS, and horizontal and vertical port scans with different anomaly intensities.

3.3.3 TES tool

TES (Traffic Entropy Spectrum) [47] tool is a Matlab tool for anomaly detection. It is developed to use Kalman filter or Karhunen-Loeve expansion as an anomaly detection method. It takes as input a set of training data, a set of evaluation data, and an anomaly vector that describes where the anomalies were injected. It provides a script for converting timeseries text files to databases. Some of the parameters specified for analysis are anomaly intensities, which have to match the ones in the anomaly vector file, PCA dimensions/components, and anomaly score thresholds. The output after running the tool is a ROC curve plot of true positives rate vs false positives rate, which includes results for different dimensions, and is based on comparing an anomaly score to each threshold.

Chapter 4

Experiments

Section 3.2 describes the process of collecting and analyzing traffic data to reach results from anomaly detection. This chapter focuses on using this process in different experimental scenarios. The experiments started with a single anomaly and a small set of traffic data, and developed into combinations of different data sets, including or not migration, and injecting anomalies of various types and intensities. At first, we wanted to see how the provided anomaly detection tool [47] reacts to the processed traffic and detects injected anomalies. The goal was to determine how well anomalies are detected for specific scenarios, and then run the same scenarios, but including migration of a service. The results allow us to compare anomaly detection with and without migration, and decide if and how it affects the final results.

4.1 Specifications

There are several experimental parameters that are changed during experiments in order to observe different scenarios. Those specifications include the set of traffic data analyzed, the type of the anomaly injected, the intensity of that anomaly, the size of migration, and if migration is included in the current experiment. After experiments are executed, the results can be observed based on anomaly score threshold and percentage of true and false positives.

The first specification in an experiment is the traffic data set used. Based on the extracted 200 services, we can choose a selection of them based on service size or protocol. For the first experiments, we start by choosing smaller sets of data to get the general idea of how the anomaly detection tool reacts to different anomaly intensities or migration. First, we sort the list of available services by flow count for the whole week of data. This data was available from the preprocessed summary file with all SWITCH services for the corresponding week. Once we can identify which service produces more traffic than others, we select the data sets:

- top10 - the first 10 services by flow count
- twentyreg - 20 relatively small in flow count services
- mixed - one top 10 service and 10 smaller ones, again based on flow count for the week (between 40 000 and 60 000 flows per 5 minutes during the day, and around 12 000 flows per 5 minutes during the night)
- alltcp - all services among the top 200 using the TCP protocol (between 600 000 and 800 000 flows per 5 minutes during the day, and around 200 000 flows per 5 minutes during the night)
- alludp - all services using the UDP protocol

Another specification is the type of anomaly and its intensity. So far we have discussed two popular types of anomalies - DDoS and port scans. Details about the generated sets of injected anomalies are presented in Section 4.2. The combination of anomaly intensity and the set of traffic data is important because if the intensity of the injected anomaly is too low, it will just

appear as part of the bursty traffic. If the intensity is too high, the anomaly will be detected at 100%, but if there are other, smaller anomalies, they will be ignored by the detection tool. With these experiments we want to explore the reaction of the detection tool at more sensitive intensities. Nevertheless, a few preliminary experiments were executed to confirm this decision. When the intensity was too low, not all DDoS anomalies were detected. First we started with the default intensity of 500 flows per 5 minutes accounting for 1% of traffic. However, when we viewed the results, they were very bad because the anomalies were hidden by the bursty traffic. Hence, higher intensities had to be generated. Empirically it was determined that the amount of anomalous flows at 50 000 flows is high enough for the datasets in question (twentyreg and mixed). However, it was not high enough for the top10 dataset. A top services generates between 3 and 30 times more traffic than a smaller one. Hence, determining the appropriate anomaly intensity size has to be done relative to the set of traffic data used.

Similar to the anomaly intensity size, the percentage of traffic data migrated affects the results of the anomaly detection. Experiments aim at finding out what is the highest percentage of traffic that can be migrated before migration is classified as an anomaly. Classification of anomalies is based on calculating anomaly scores, which represent how far the current point is from the definition of normal traffic, and comparing them to a predefined threshold. In our case this point is a timeseries entry for specific 5 minutes. When the score is higher than the threshold, it is classified as an anomaly. Each run of an experiment does the following: calculate an anomaly score for each point, compare this score to a threshold, decide whether an anomaly was successfully detected or not, and, based on scores for all points forming the dataset, calculate the TPR and FPR for the dataset. Then, repeat these steps for each threshold in the predefined list (0 0.001 0.01 0.1:0.05:10 10:10:100). When the TPR vs FPR results are plotted for each of these thresholds, they form a ROC curve. It can be used to find the threshold which gives a desired balance between TPR and FPR. This process is then completed several times for different number of components used by PCA to reduce the dataset dimensionality. The number of components used is denoted by k and a ROC curve for each k is plotted to produce the final result of an experiment. This result can show how the anomaly detection tool performs based on different thresholds and components used in analysis.

4.2 Injected Anomalies

Table 4.1 summarizes the anomalies that were injected with the help of Flame [23]. They vary in type and intensity, and different models are used depending on the type of anomaly and the subset of data, where it is injected. Models of different anomaly types differ in the distribution of IP addresses and port numbers, depending on the definition of the anomaly. Different models of the same anomaly type only differ by destination IP address (service under attack). The anomaly intensity is represented by a number, which is a fraction of a total count of anomalous flows set to 5 million. Table 4.1 shows both the number used by Flame during anomaly generation, and the more meaningful count of anomalous flows for a 5 minute injection. The sets of anomalies were injected at random timestamps in traffic traces. Each experiment always includes a single type of anomaly with multiple injections. These sets are given names that correspond to the name of the directories where they are stored on the cluster, used for these experiments. The different models are also stored there with their corresponding names.

As mentioned before, there are two types of anomalies investigated in this work. They were modeled using Flame by supplying values to required variables.

Distributed Denial of Service (model01 to model06):
Transport protocol: TCP
Source IP addresses: a list of randomly generated external addresses
Destination IP addresses: an internal address belonging to the set of traffic data
Source port numbers: randomly chosen port
Destination port numbers: the port for the service with destination IP address
Flow sizes: 2 packets per flow and between 1 and 1024 bytes

Name of anomaly set	Type of anomaly	Injection model	Intensity	Anomalous flows per 5 min
ddos	DDoS	model01	0.0001	500
ddos2	DDoS	model03	0.001	5 000
ddos3	DDoS	model03	0.01	50 000
ddos4	DDoS	model04	0.1	500 000
ddos6	DDoS	model05	0.004	20 000
ddos7	DDoS	model05	0.002	10 000
ddos8	DDoS	model06	0.03	150 000
ddos9	DDoS	model06	0.05	250 000
portscan1	vertical port scan	model07	0.001	5 000
portscan2	vertical port scan	model07	0.004	20 000
portscan3	vertical port scan	model07	0.01	50 000
portscan4	vertical port scan	model07	0.03	150 000
portscan5	horizontal port scan	model08	0.001	5 000
portscan6	horizontal port scan	model08	0.004	20 000
portscan7	horizontal port scan	model08	0.01	50 000
portscan8	horizontal port scan	model08	0.03	150 000

Table 4.1: Summary of anomaly sets

Vertical port scan (model07):
Transport protocol: TCP
Source IP addresses: a random external address
Destination IP addresses: an internal address belonging to the set of traffic data
Source port numbers: randomly chosen port
Destination port numbers: a randomly generated list of ports
Flow sizes: 1 to 2 packets per flow and between 1 and 512 bytes

Horizontal port scan (model08):
Transport protocol: TCP
Source IP addresses: a random external address
Destination IP addresses: an internal address belonging to the set of traffic data
Source port numbers: randomly chosen port
Destination port numbers: port 80
Flow sizes: 1 to 2 packets per flow and between 1 and 512 bytes

In order to understand the results of the executed experiments better and the conclusions based on them, it is necessary to understand how traffic behaves with and without migration, and with and without anomalies for different traffic features. Since anomalies are injected based on flow counts, we start by observing the trace for flow counts for data without injected anomalies. Figure 4.1 shows the baseline traffic for the flow counts. It is visible that traffic is bursty, and perhaps already includes anomalies, which were not injected manually. In addition, there are less flows during the night compared to traffic during the day. Figure 4.2 shows a traffic trace of the same data and also based on flow counts. The difference is that this trace includes migration and anomalies. There are two things to notice in the figure. First, migration is visible to occur between the third and forth day, and overall traffic decreases, but traffic pattern is not changed. It is also interesting to notice where the anomalies are. Due to the bursty traffic and lower anomaly intensity, anomalies are not easy to distinguish by the naked eye. On the other hand, anomalies in the trace shown in figure 4.3 are clearly visible because their intensity is higher with respect to the traffic. This is a reason why experiments are conducted both with high and low intensities in order to observe how the detection system reacts to these situations.

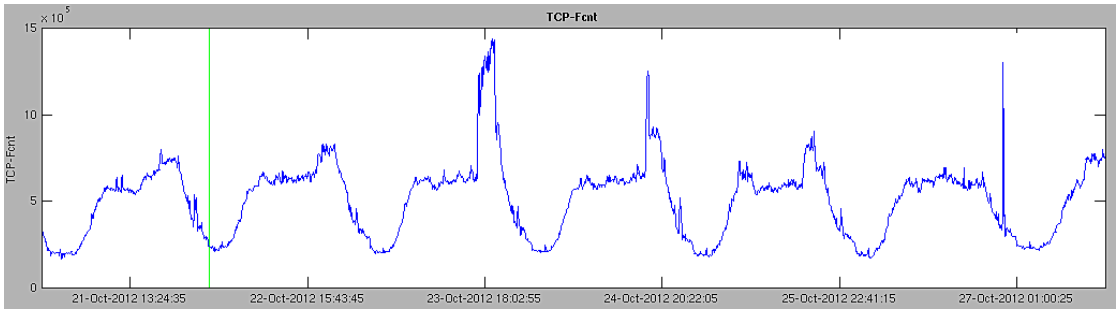


Figure 4.1: Baseline traffic based on flow counts

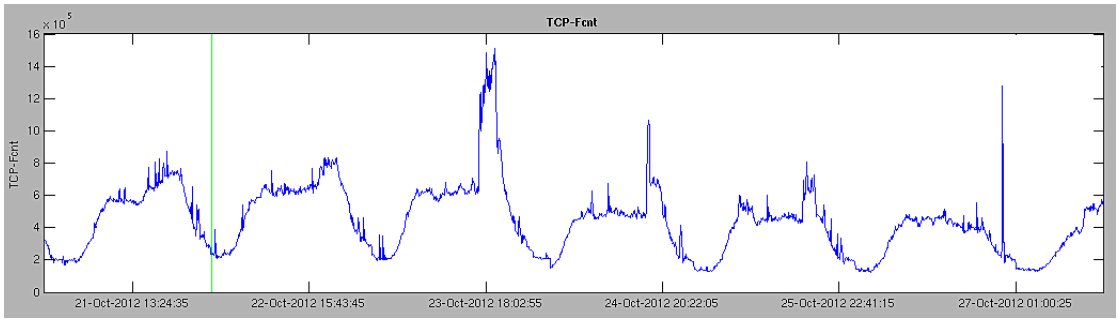


Figure 4.2: Migration traffic based on flow counts

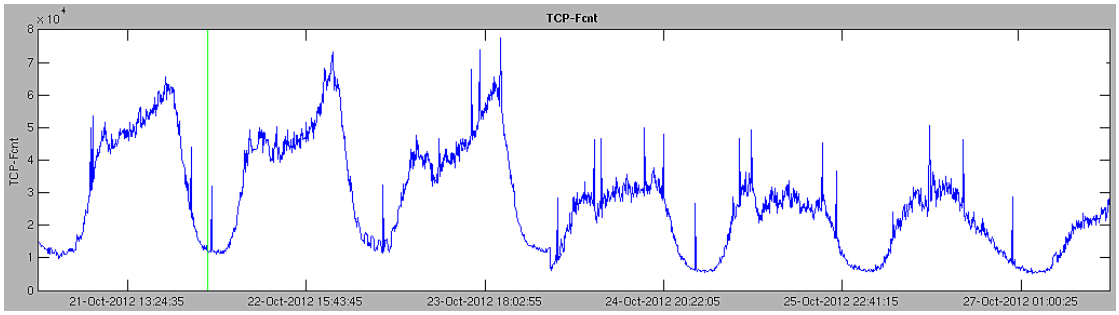


Figure 4.3: Traffic trace with migration and high anomaly intensity based on flow counts

In addition to anomaly intensity, it is also interesting to observe how different traffic features react to anomalies and to migration. Figure 4.4 shows the baseline traffic based on destination port entropy, and figure 4.5 shows the same traffic trace with migration and anomalies. These are results from the same experiment as the results presented in figure 4.2. We can see that migration is not easily seen with this traffic feature, whereas the injected anomaly (vertical port scan) is much more intense than it is with flow counts. This is due to the characteristic of the anomaly to attack a wide range of port numbers. Thus, it widens the range of destination port numbers in the trace and affects the entropy value. Figure 4.6 shows a different trace with migration and an injected anomaly (DDoS) where migration is more distinguishable. The interesting part of this figure is that the anomaly intensity seems to increase compared to the overall traffic when the intensity of anomalies injected before and after the migration is the same. Thus, we can say that migration might even help anomaly detection.

4.3 List of Experiments

Tables A.1, A.2, A.3 in Appendix A list a detailed description of experiments ran to determine the affect of migration on anomaly detection. Prior to these, preliminary experiments were conducted to determine the subsets of data and anomaly intensities. The following scenarios only focus on two groups of traffic data - mixed and alltcp. Mixed is a smaller subset of data,

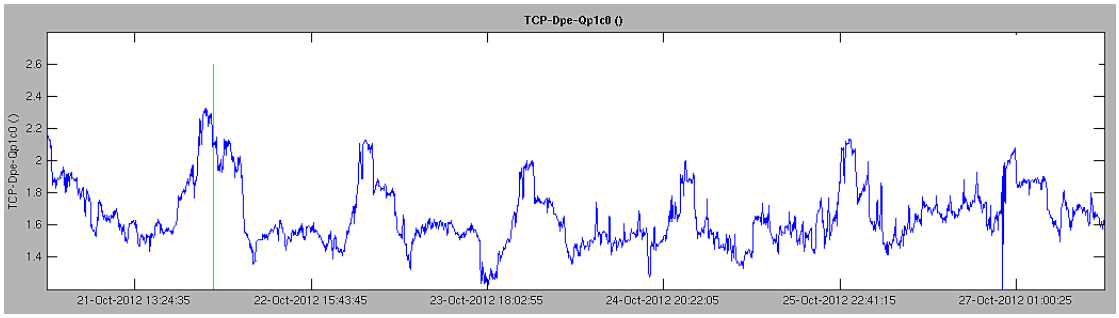


Figure 4.4: Baseline traffic based on destination port entropy

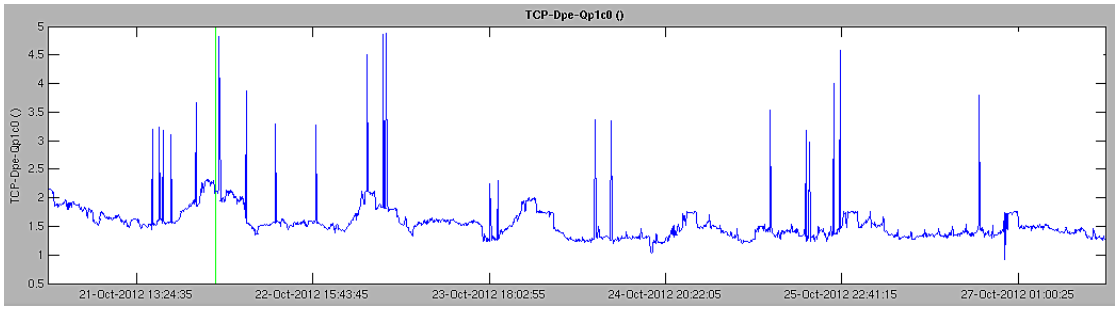


Figure 4.5: Migration traffic based on destination port entropy

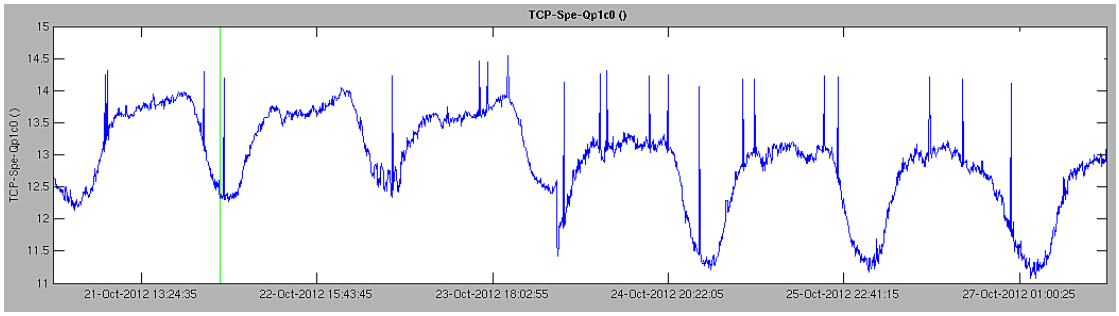


Figure 4.6: Traffic trace with migration based on source port entropy

mostly used to develop an idea of how the system reacts to a situation such as increase in anomaly intensity, inclusion of migration and migrating different amount of data. Then a hypothesis can be made about what to expect on a bigger set of data, alltcp, which includes all TCP traffic from the processed data. Each experiment is labeled with an ID in the order of execution to keep track of results. For each set of data, a baseline timeseries is generated. It is used for training data. The evaluation data is different in every experiment, depending on type and intensity of anomaly injection, and also on migration. Hence, the table also includes the name for the generated anomalous timeseries. Two of the other columns are the name of the set of anomalies and the anomaly intensity. The last column covers migration. If the value is "no", then there were just anomalies injected in that scenario and no migration was included. When the value is a percentage, that percentage describes how much of the total traffic data was migrated. There is one more possibility for migration - "special". This was a single experiment, which tried to train traffic data with migration included. The results are described in the next chapter.

Here is a summary of all experiments:

- *mixed* dataset
  - DDoS anomalies at intensities 0.001 and 0.004 with migration of 0%, 5.8% and 30% of overall traffic

  - Port scan anomalies at intensities 0.001 and 0.004 with migration of 0%, 5.8% and 30% of overall traffic
  - Special - trained with migration; tested with DDoS anomalies

- *alltcp* dataset
  - DDoS anomalies at intensities 0.03 and 0.05 with migration of 0%, 0.27%, 2.5%, 10.64%, 21.4%, 28%, and 39.7% of overall traffic
  - Port scan anomalies at intensities 0.01 and 0.03 with migration of 0%, 2.5%, 10.64%, 21.4%, 28%, and 39.7% of overall traffic

Chapter 5

Results and Analysis

This chapter shows anomaly detection results for the experiments described in Chapter 4. These results present how virtual service migration affects anomaly detection accuracy at different anomaly intensities for different types of anomalies and for different migration size. We start by evaluating results based on type of anomaly. All results are presented by True Positive Ratio (TPR) and False Positive Ratio (FPR) for each anomaly score threshold. TPR needs to be high and FPR needs to have low values. These two rates are related to each other using the Receiver Operating Characteristic (ROC) curve. ROC curve is a plot of true positives vs false positives, which illustrates the performance of a system by comparing the two rates.

For these experiments we focus on two traffic datasets - *mixed* and *alltcp*. We observe anomaly detection of attacks injected at different anomaly intensities. Then, we include migration to the same scenario to observe the effect it has on the final results. The reason for using two different datasets is that we want to observe results on the first dataset, make conclusions, and then test if the same conclusions apply to a bigger dataset. If the hypothesis we form for the smaller dataset is valid for the bigger dataset, then we could induce that the hypothesis we prove for the bigger dataset will be valid for a complete cloud data center traffic. The goal is to show if and how migration affects anomaly detection accuracy in a cloud data center without having access to a real-time cloud traffic. The variation of anomaly intensity is also important. Results from experiments will show that with a higher anomaly intensity the sensitivity of the anomaly detection towards migration decreases. This could be a result which concludes that migration has no affect on anomaly detection. However, if the injected anomalies are smaller in size, the sensitivity of the detection tool increases, and migration can be classified as an anomaly. Such results will show that virtual service migration has a higher influence of anomaly detection accuracy that results with higher anomaly intensity. The reason we are using more than one size of anomaly sets is to not base our conclusions on a single case. If we show that migration has an affect on anomaly detection on different sets of data with different anomaly sizes, then the results will be more conclusive than if they were based on a single experiment, or a single variation of a scenario.

Once we have observed how virtual service migration can affect anomaly detection depending on the anomaly intensity and the amount of traffic migrated for DDoS anomalies (volume based attacks), we can conduct the same experiments for a different anomaly type - port scans. Port scans are non-volume based attacks, and it is possible migration has a different effect than for DDoS attacks. We focus on vertical and horizontal port scans. Vertical port scans are characterized by a dominant destination IP address and requests send to multiple ports on that address. Horizontal port scans target multiple IP addresses, but only scan for a single specified port.

A total of 58 experiments were executed. However, to avoid redundancy this chapter only discusses some of those results in order to present the reached conclusions. Tables that summarize all results are included in Appendix A, in addition to a plot of results for each experiment, added in Appendix B.

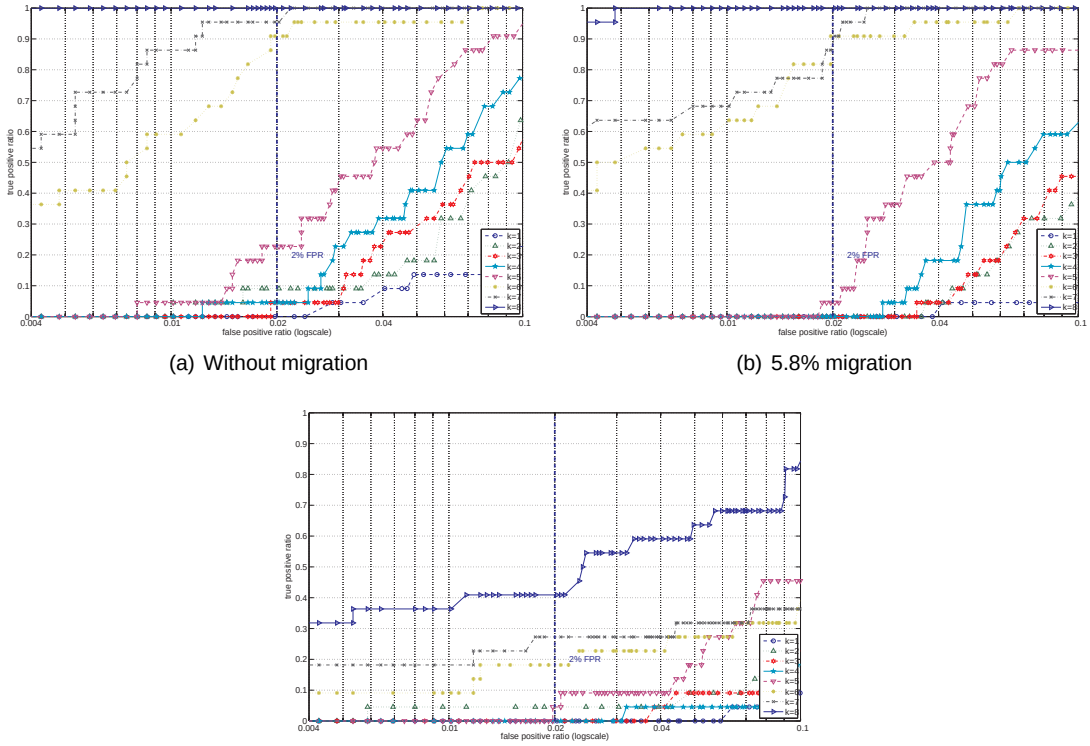
5.1 Dataset *mixed*

We first conduct experiments on a smaller set of data, which we call *mixed* dataset. It consists of one top service, which accounts for 30% of the traffic in this dataset, and 10 smaller services, which account for the rest of the traffic. We choose two anomaly intensities - 0.001 and 0.004, determined empirically, and we observe results from two anomaly types - DDoS and port scans. The reason for using a smaller subset of data is to reach some conclusions at the end of these experiments, and then form a hypothesis for a bigger set of data, which we can then test with the *alltcp* dataset.

5.1.1 Distributed Denial of Service

Anomaly intensity 0.001

We start by exploring the *mixed* dataset with a DDoS attack of intensity 0.001. This means there are 5 000 anomalous flows for a 5 minute range. The normal traffic dataset contains between 40 000 and 60 000 flows per 5 minutes on average. This means that the traffic amount will increase by 10% when anomalies are injected. Hence, they should be detected successfully. Figure 5.1 (a) shows the ROC curves for this scenario for detecting anomalies in normal traffic without migration involved. We can notice that for this anomaly intensity the TPR is above 90% only for $k = 6, 7, 8$. The traffic is bursty and the detection tool is sensitive to the chosen anomaly intensity. We would need to conduct the same experiment with higher intensity, but first we want to observe how migration will affect the current results, nevertheless.



(a) Without migration

(b) 5.8% migration

(c) 30% migration

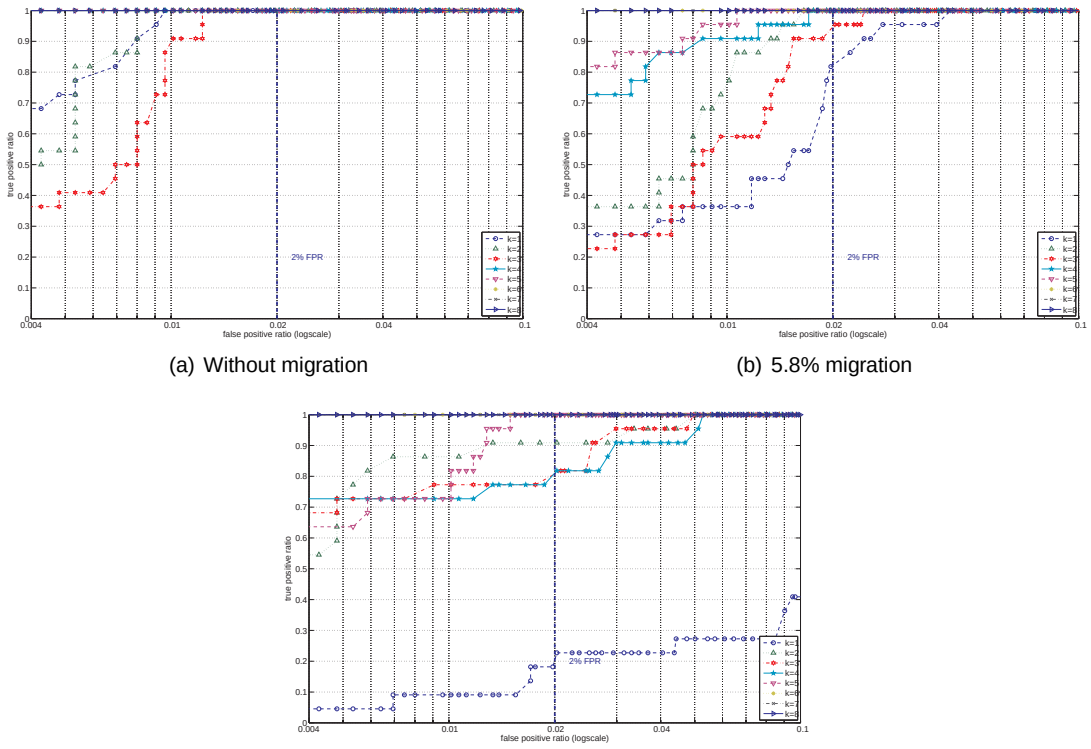
Figure 5.1: Anomaly detection results for *mixed* dataset for DDoS attacks with intensity 0.001

Figure 5.1(b) shows the ROC curves for the same scenario, but with a migration of a small service included. That service accounts for 5.8% of the total traffic. Compared to the results without migration, the TPR for $k = 6, 7, 8$ is still above 90%. However, the TPR for less dimensions has decreased. For example, for $k = 5$ the TPR has decreased from 22% to 5%, and for $k < 5$ it has gone down to 0%. Thus, depending on the desired FPR and on the

number of migrations, migration of 5.8% of the traffic visibly affects the anomaly detection, but anomalies are detected with a very similar accuracy as without migration.

The next experiment assumes the same scenario, but this time the migration accounts for 30% of overall traffic. This is a significant amount, and perhaps it is an exceptional case that one third of all data center traffic will be migrated, but it is still interesting to observe how the anomaly detection reacts. This situation can arise from a large-scale failure, for example, caused by a natural disaster or a significant power outage. Figure 5.1(c) presents the results. Even the higher dimensions do not give satisfying results. Anomaly detection accuracy has decreased a lot, and the highest TPR now is 40%. Hence, we can conclude that migrating a very large percentage of overall traffic will result in a lot of false alarms. If somebody wants to migrate a large portion of their cloud data center, the detection system should be informed about it, so it can adapt to the large change in traffic.

The behavior of the detection system can be explained by the changes migration causes to the evaluation timeseries. The flow, packet, and byte counts decrease, and the entropy also changes since the distribution of IP addresses is now different. When the difference between the baseline timeseries and the timeseries including migration is calculated, the resulting anomaly scores must be above the threshold if they affect the true and false positive rate. When more points have anomaly score higher than the threshold, and they are not defined as anomalies, they are counted towards the false positives and not towards the true positives as for the traffic without migration. Then the point on the plot, which represents the current threshold for the specific number of dimensions used, moves down and to the right. When rates for all thresholds are calculated, if most of them have moved down and to the right, it appears as if the ROC curve has shifted to the right, as it has in the described experiments.



(a) Without migration (b) 5.8% migration

(c) 30% migration

Figure 5.2: Anomaly detection results for *mixed* dataset for DDoS attacks with intensity 0.004

Anomaly intensity 0.004

Next, we decide to explore a higher anomaly intensity, injecting 20 000 anomalous flows per 5 minutes. Compared to the normal traffic dataset, which contains between 40 000 and 60

000 flows per 5 minutes on average, the anomalies will account for one third to one quarter of the traffic. The first experiment with this set up injects anomalies without including migration. This will be the base for comparison later when migration is included. Figure 5.2(a) shows the results of the anomaly detection. For all k, the TPR is 100% for 2% FPR. This means that for all dimensions, most anomalies were detected successfully. For FPR below 1%, the first three dimensions give lower TPR values. In figure 5.2(b), we can observe that these lower values have shifted to the right, and now the TPR is between 80% and 100% for 2% FPR. These are the results after a small service migration of around 5.8% of all traffic. After a larger migration of 30% of overall traffic, the effect on anomaly detection is shown on figure 5.2(c). The result for $k = 1$ has decreased from 100% to below 30%. Values for TPR for $k = 2, 3, 4$ are now between 80% and 90%. These results show that migration causes anomaly detection results to become less accurate, just as the results for the lower anomaly intensity showed.

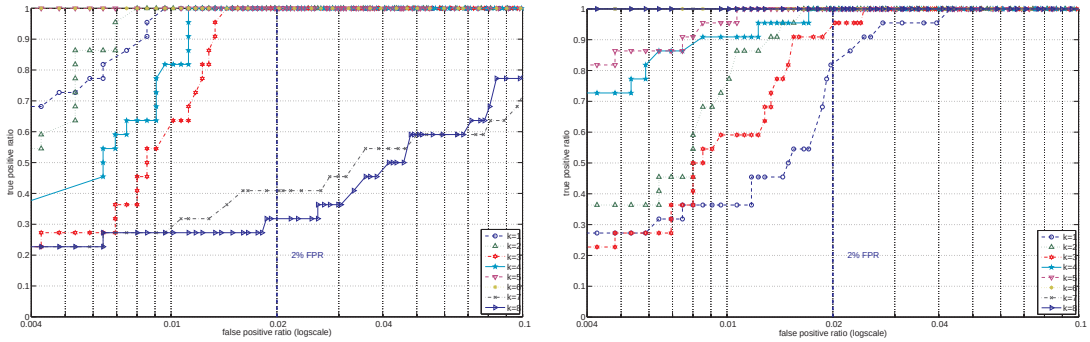
It is interesting to observe the difference between the results of the different anomaly intensity sizes. The first difference is the detection accuracy expressed by TPR and FPR values. For the higher intensity, the TPR is 100% compared to 20% or 90% for the lower intensity. This is true both for experiments without migration, and for experiments with migration. However, we want to observe how migration affects these two experiments differently. We choose to compare values for $k = 1$ for experiment 5, and $k = 5/6/7$ for experiment 4. These dimensionalities were chosen because they show the worst drop of TPR percentage. The first set of experiments gives a drop of up to 17% for the smaller service migration, and a drop of 60 to 70% for dimensionalities 6/7/8 for the large service migration. The second set of experiments shows a drop of 20% for the smaller service migration, and 80% for the large service migration. However, this decline is only valid for one dimensionality. The decrease of TPR percentage might be slightly larger for the higher intensity, but it is only applicable to the first dimensionality, whereas migration in the first set of experiments affects results for more dimensionalities. Hence, the affect of migration with lower anomaly intensity is stronger than for higher anomaly intensity traffic.

Training migration

Observing that migration has an affect on anomaly detection according to the first experiments, we want to try a different approach of training and evaluating data before continuing with further experiments. The idea is to train normal traffic to include migration already. Thus, once a migration occurs in the evaluation, the detection tool should recognize it as part of normal behavior and not detect it as an anomaly. We first ran an experiment by creating a baseline timeseries, which includes a migration of a small service accounting for 6% of overall traffic. Then we generated the anomalous timeseries, which does not include any migration, but just the usual anomaly injections. Figure 5.3(a) shows the results produced by the anomaly detection tool. For the two highest dimensions, the TPR for 2% FPR is between 30% and 40%. For the others it is 100%. Compared to figure 5.2(a), which shows the results for the same anomaly type and intensity without including migration and has 100% TPR, the current model is performing worse. However, it is more suitable to compare it to the model that is trained without migration, but it is tested with migration. In this way, we are comparing experiments, which both have migration of a similar size involved. TPRs for the experiment in figure 5.2(b) are all higher than 80%, which is better than the 30% produced with training migration. Thus, we can conclude that including migration in the training of normal traffic data does not improve anomaly detection results, and will not be pursued further. It is not a solution to the problem if anomaly detection accuracy decreases when migration is involved in traffic traces. Training migration to be part of baseline traffic results in more bursty traffic than without migration, and when the baseline model is created, it has to widen in order to include traffic before and after migration. Then this model will be less sensitive and might not detect anomalies as it should. It is possible that this is the reason the anomaly detection results are not good after training with migration.

5.1.2 Vertical Port Scan

We continue by observing the results for a vertical port scan with injected anomalies at intensity 0.001. Figure 5.4(a) shows the ROC curve for anomaly detection without migration. For 2% FPR, TPR is 100% for all dimensionalities. For less than 1% FPR, the first three dimensionalities hold



(a) Detection after training with migration

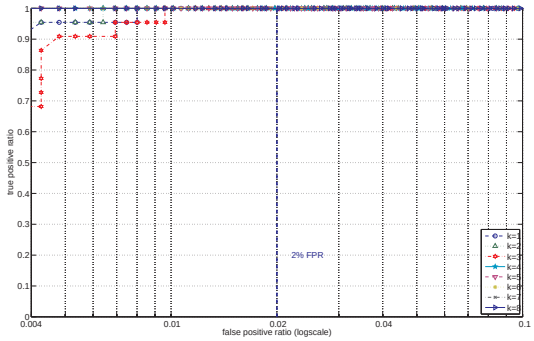
(b) Detection with normal traffic at 5.8% migration

Figure 5.3: Anomaly detection results for training normal traffic with migration

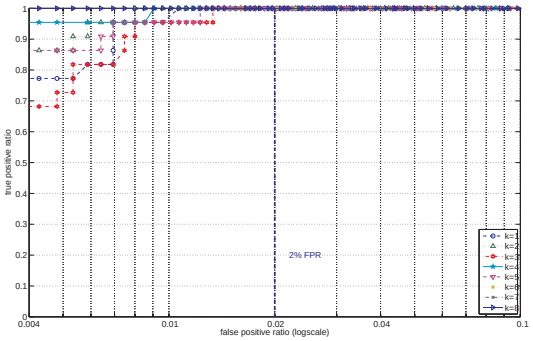
TPR values between 90% and 100%. Usually, we only look at the decided value for FPR - 2%, but in this case the results show no change at that rate, but still show changes for lower rates. This means that there is just a small number of points that have an anomaly score above the first few thresholds, and everything else that gives an anomaly score above 0 is an injected anomaly. It is interesting to notice that vertical port scans are much more accurately detected for this intensity than the DDoS attacks were. The reason is probably the nature of the anomaly - port scans are non-volume based anomalies and are more visible in IP and port distribution entropies than in flow or packet counts. Figure 5.4(b) is the plot for the same scenario, but with migration of a smaller service accounting for 5.8% of the traffic. There is no change for FPR of 2%, but it can be noticed that for FPR below 1% now there are more dimensionalities that do not result in 100% TPR. This means that some of the points that include migration have an anomaly score higher than 0, but still a low enough score to be below most of the defined thresholds. Thus, the results still show very accurate anomaly detection even with migration involved. The next experiment includes a 30% migration, figure 5.4(c), and as expected has a stronger effect on anomaly detection. However, it is not as disruptive as the one for DDoS anomalies. The ROC curve for $k = 1$ has changed and now for 2% FPR the TPR is 64% compared to 100% before. However, the ROC curves for the higher dimensions stay at 100% TPR. These results are for migrating 30% of the traffic, and if more than one dimension is used in the analysis, then migration would not affect the final anomaly detection results. Since results for anomaly intensity of 0.001 are already very good, it is expected that for a higher anomaly intensity migration will cause no decrease in anomaly detection accuracy. This is confirmed by plots in figure 5.5, which all show 100% TPR for any dimensionality and any FPR even when 30% of traffic is migrated.

5.1.3 Horizontal Port Scan

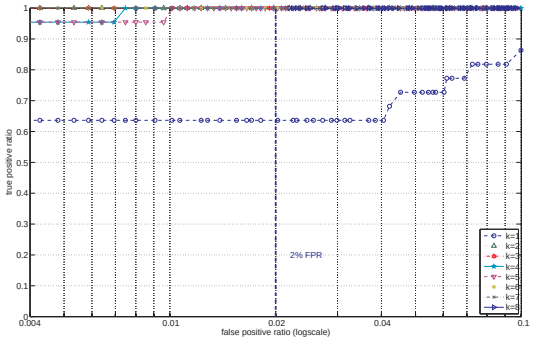
Next, we conduct the same experiments but with a horizontal port scan. The difference will be that in a horizontal port scan there is a dominant destination port, but no dominant destination IP address. Thus, there will be no high concentration of requests on any specific destination service. Instead, the IP address distribution will widen. First, we conduct an experiment without migration at anomaly intensity of 0.001 in order to set a baseline for comparison once migration is introduced to the trace. Figure 5.6(a) shows this baseline. Anomalies are detected with TPR of 77%, 95%, and 100% for $k = 1$, $k = 2$, and $k >= 3$ respectively. Then migration is included, starting with migrating less than 6% of overall traffic, figure 5.6(b). The ROC curves have shifted slightly to the right, and now the TPR for 2% FPR is 72% and 90% for the first two dimensionalities. Hence, the accuracy of the anomaly detection has decreased, and migration of services has an effect. With vertical port scans, anomalies were detected with better TPR and migration did not affect results. However, horizontal port scans are characterized by different features, and they are not as well combined with migration as the ones for vertical port scans. We continue with an experiment with 30% migrated traffic. Since we have already observed an effect on anomaly detection by migration for a lower percentage of migrated traffic, we expect that these results will show even greater impact of migration. Figure 5.6(c) plots the resulting ROC curves. The TPR for the first dimensionality drops down to 40%. It is interesting



(a) Without migration

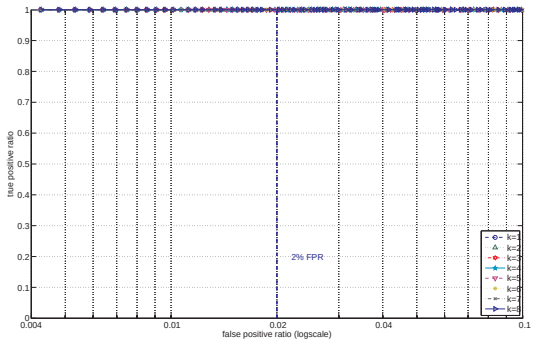


(b) 5.8% migration

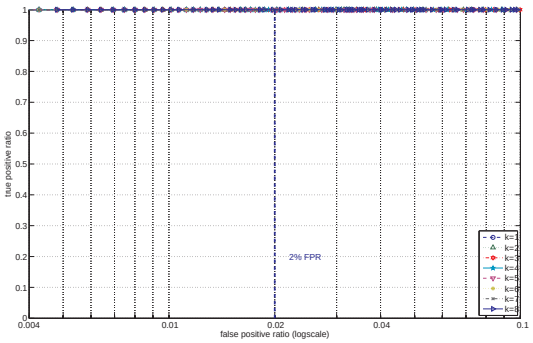


(c) 30% migration

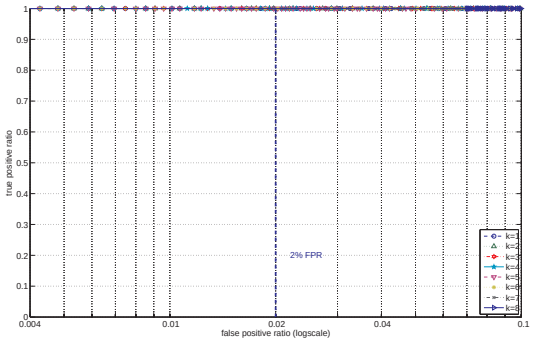
Figure 5.4: Anomaly detection results for *mixed* dataset for vertical port scan attacks with intensity 0.001



(a) Without migration



(b) 5.8% migration

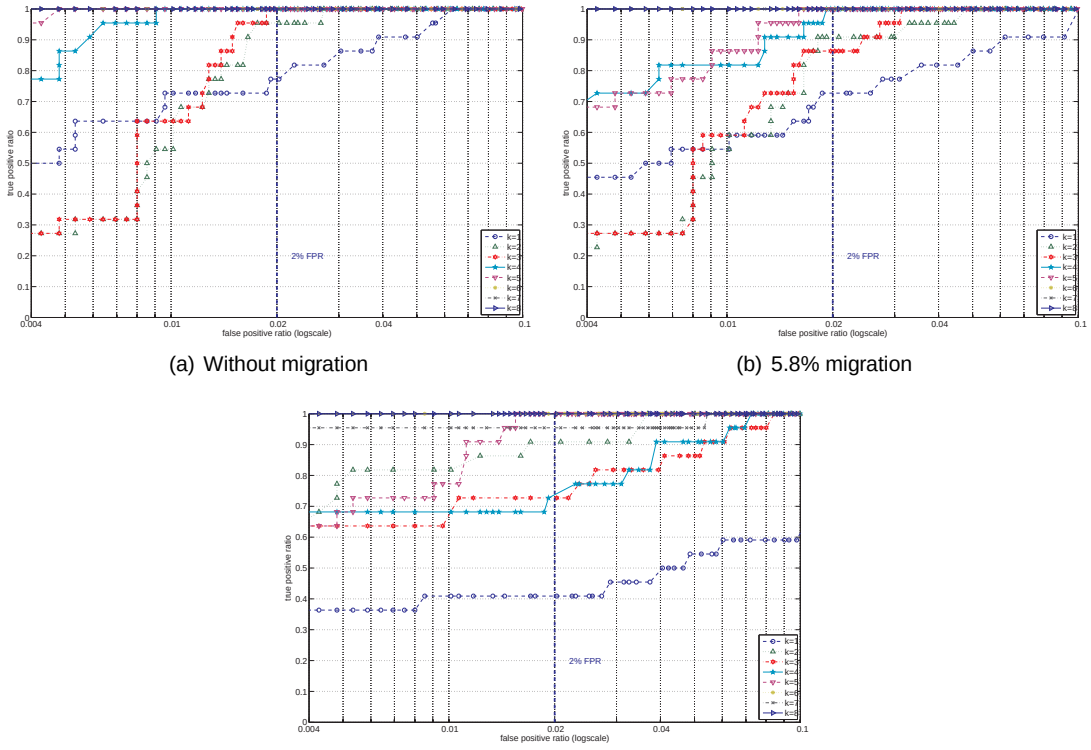


(c) 30% migration

Figure 5.5: Anomaly detection results for *mixed* dataset for vertical port scan attacks with intensity 0.004

that it follows a similar pattern as the TPR for the first dimensionality in figure 5.4(c), which

can be described as flat and mild ranging only between 37% and 60% for FPR up to 10%. The ROC curve for $k = 1$ in figure 5.4(c) ranges between 64% and 88%, also a bit more than 20% difference between its values. This is interesting because most of the other curves span larger ranges and are not as far from the other curves as the ones for dimensionality 1. This flatness of the ROC curve is a result of anomaly scores based on thresholds. If the TPR for different thresholds are similar to each other, this means that there are just a few or no points that have an anomaly score between these thresholds. Hence, by using just one dimensionality for calculating results in those two cases, the anomaly scores are either very high and always above a certain threshold, or they are below all thresholds and are classified as normal traffic. Looking at figure 5.4(c), we can notice that the TPR stays the same for approximately three quarters of the thresholds. This means that migration produces high anomaly scores that are above most of the thresholds. That's why these points are classified as anomalies. However, when more components are used to analyze the traffic, these anomaly scores decrease. The TPR for the other dimensionalities for the horizontal port scans experiments has also decreased, for example, down to 73% for two of the other dimensionalities. Hence, as expected, migration affects the traffic features enough to affect the anomaly detection as well.



(a) Without migration

(b) 5.8% migration

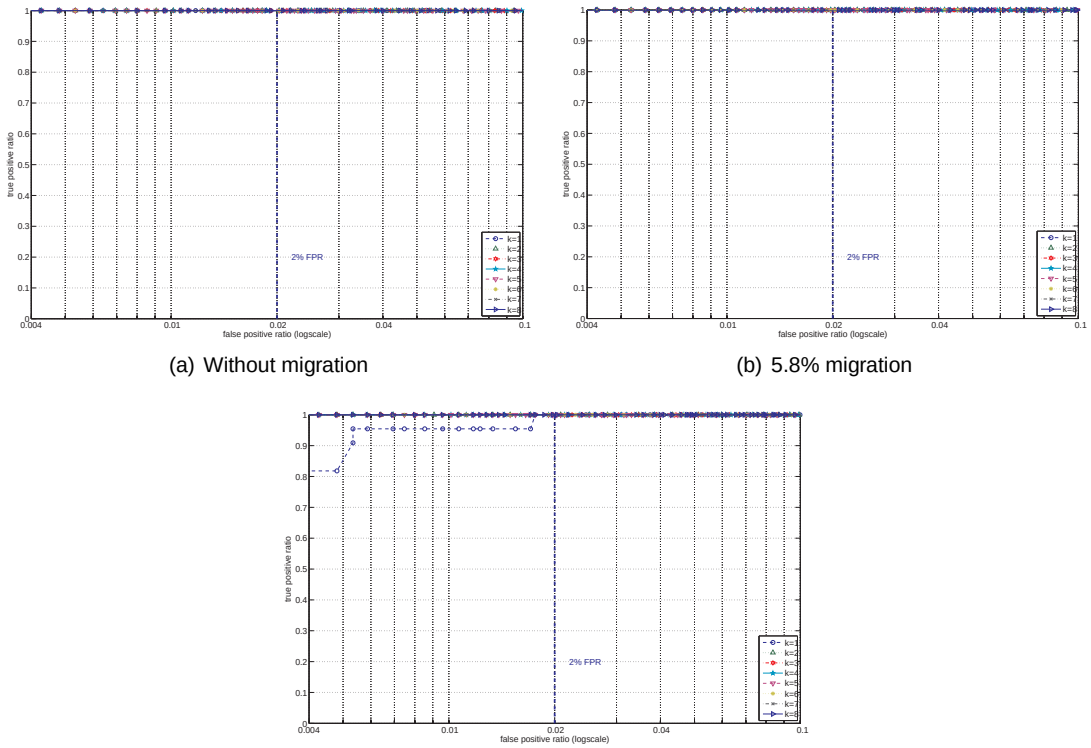
(c) 30% migration

Figure 5.6: Anomaly detection results for *mixed* dataset for horizontal port scan attacks with intensity 0.001

Next we conduct the same experiments with higher rate of anomaly injection - 0.004. Figure 5.7(a) shows the 100% TPR for anomaly detection without migration. For a small migration, the results are identical, figure 5.7(b). For the 30% migrated traffic results also show 100% TPR for all dimensions, but there is a small decrease for the first dimensionality for very low values of FPR. This means that migration at a few points results in low anomaly scores. Hence, the effect of migration on anomalous traffic with horizontal and also for vertical port scans at intensity 0.004 is negligible.

5.1.4 Conclusions

Before we continue with experiments on the bigger set of traffic data we need to make some conclusions based on the results from the *mixed* dataset. We have observed that the type of



(a) Without migration

(b) 5.8% migration

(c) 30% migration

Figure 5.7: Anomaly detection results for *mixed* dataset for horizontal port scan attacks with intensity 0.004

anomaly injected into the traffic affects how much migration changes anomaly detection results. For DDoS attacks, migration receives a high enough anomaly score to be classified as an anomaly. In contrast, when the injected anomalies are vertical port scans, migration is not detected as an anomaly by the detection system. Judging by figure 4.6, migration actually intensifies the vertical port scan anomalies. The only scenarios when migration points receive very high anomaly scores and are classified as an anomaly is at a lower anomaly intensity combined with high percentage of migrating traffic, figure 5.4(c). Experiments with horizontal port scans, on the other hand, have shown similar results to the ones from experiments with DDoS anomalies. Even though horizontal port scan anomalies are detected with higher rates for the same intensity by the anomaly detection tool, in both cases migration affects the final results.

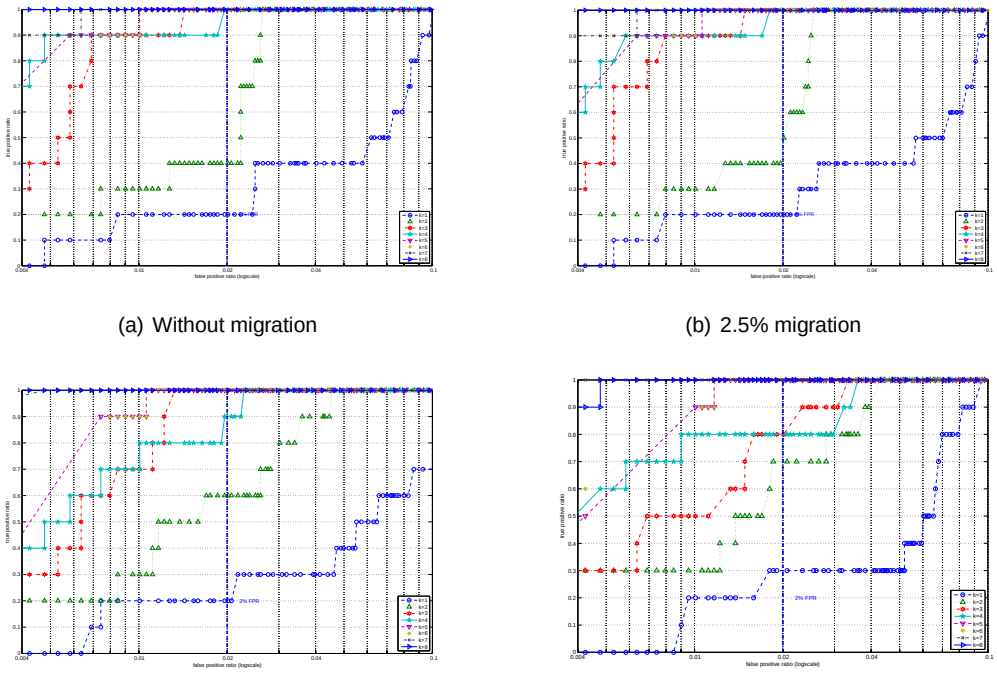
5.2 Dataset *alltcp*

After the experiments conducted on the *mixed* dataset so far, we can make a few conclusions, and then observe if they will apply to the *alltcp* traffic. The expectations are that for a higher anomaly intensity the affect on anomaly detection will be smaller than the one on a dataset injected with anomalies of smaller size. We also expect that migration of a larger size will result in more false positives than migration of a smaller size. Anomaly type will also affect final results. We expect that for DDoS attacks we will get a lower percentage of traffic migrated before migrations starts disrupting anomaly detection results, than for horizontal or vertical port scans. The highest percentage limit of traffic migrated should be for vertical port scans. In addition, the effect of a top service migrated will not be as large as the effect of the same service on the *mixed* dataset because the *alltcp* dataset consists of more flows. To be exact, the *alltcp* dataset is composed of 16 times more flows than the *mixed* dataset. The size of the dataset also requires different anomaly intensities. The anomaly intensities corresponding to the ones used for the smaller set of traffic data are $0.001 * 16 = 0.016$ and $0.004 * 16 = 0.064$. The anomalies injected with the lower intensity, however, were not well detected by the anomaly detection tool. Moreover, the anomalies injected with the higher intensity were too strong for the detection tool to focus on the migration traffic. Hence, for the next experiments we will choose anomaly in-

tensities between the ones corresponding to the anomaly intensities for the *mixed* dataset. We choose to observe results with anomalies injected with intensities 0.03 and 0.05. In addition, we have observed that port scans are detected with higher accuracy than DDoS attacks. Hence, for the port scan experiments we will use slightly lower anomaly intensities than the ones used for the DDoS injections - 0.01 and 0.03.

5.2.1 Distributed Denial of Service

Here we present the results for DDoS anomaly detection with migration of different sizes. We have performed a lot of experiments with different migration traffic percentage, but here we just present a few of them to summarize the findings. All of the results can be viewed in the Appendix.



(a) Without migration

(b) 2.5% migration

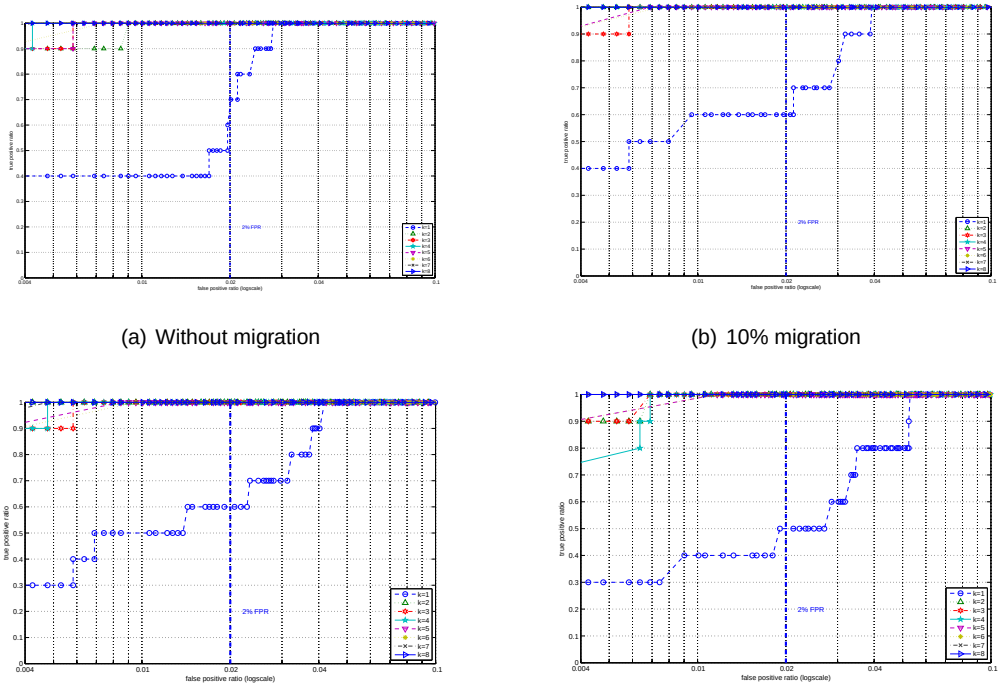
(c) 10% migration

(d) 28% migration

Figure 5.8: Anomaly detection results for *alltcp* dataset for DDoS attacks with intensity 0.03

The first anomaly intensity we use to inject anomalies in the traffic trace is 0.03. This means an attack consists of 150 000 anomalous flows injected in 600 000 to 800 000 flows per 5 minutes. Hence, the anomalies will account for 15% to 20% of the traffc at the moment of injection. Figure 5.8(a) shows the results from anomaly detection on an evaluation set of data without migration. Anomalies are detected with 100% for $k = 3$ or higher. For the lower dimensionalities of the analysis model, the TPR for 2% FPR is 20% and 40% for $k = 1$ and $k = 2$ respectively. Next, we add the migration component. We start by migrating 2.5% of the traffic, figure 5.8(b). The resulting ROC curve looks a lot like the one for the baseline traffic, and the TPR values for all k are the same. However, this does not necessarily mean migration had no effect on the anomaly detection. If we look closer at the ROC curve for 2 dimensions ($k = 2$), we can find out the threshold for which the FPR is 2% and TPR is 40%. In figure 5.8(a), it is the 27th point on the curve, which gives it a threshold of 1.25 for anomaly scores. In figure 5.8(b), the point for 2% FPR is the 23rd point, corresponding to threshold of 1.05. The rates for the original threshold are now 2.3% FPR and 60% TPR, which actually shows improvement in anomaly detection. However, if we look at the results for more dimensions, for example 4, there is no change in the threshold which gives 2% FPR and 100% TPR. Naturally, when more dimensions are used, there is more information to help the detection tool analyze traffic and determine anomaly scores. Hence, we should not base our decision on changes for just

one dimensionality. Since for 4 dimensions there was no change in the anomaly detection, we conclude that at this rate of migration it does not affect the final results. We continue by exploring 10% traffic migration, which is presented in figure 5.8(c). The original threshold for 2 dimensions is at 60% TPR, which is better than without migration. However, the ROC curve for $k = 4$ has moved to 90% TPR for 2% FPR, which already gives a significant change in the anomaly detection results. It means that there are points which scored higher anomaly scores than the threshold corresponding to 2% FPR. If we increase the amount of traffic migrated even more (to 28%), the affect on the final results is even more visible, figure 5.8(d). Hence for anomaly intensity of 0.03 and DDoS anomaly type, we can conclude that migration already changes anomaly detection results at 10% migrated traffic. At lower rates of migration it can remain undetected and even boost detection of existing anomalies. This could occur because, when there is less traffic, anomalies become a higher percentage of the traffic or because the anomalies become more visible in the address and port distribution when this distribution is affected by removing several services. It could also be due to removing bursty traffic, which was part of the migrated services. Perhaps this traffic was classified as anomalies before, and once it is gone, the number of falsely labeled anomalies decreases.



(a) Without migration

(b) 10% migration

(c) 21% migration

(d) 39% migration

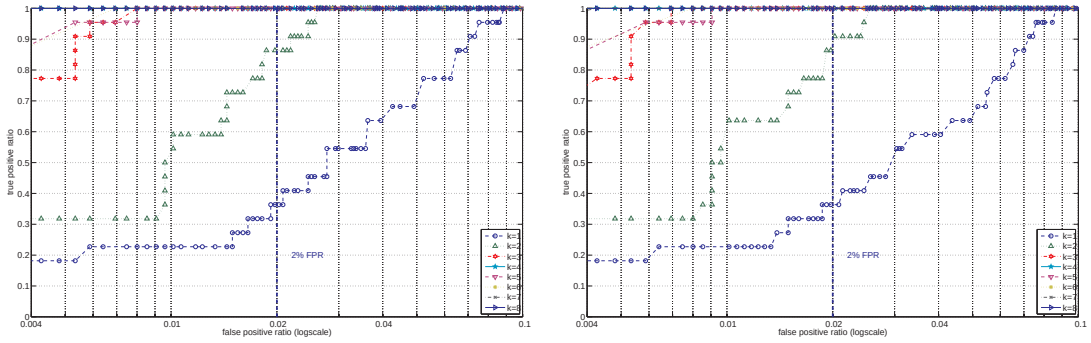
Figure 5.9: Anomaly detection results for *alltcp* dataset for DDoS attacks with intensity 0.05

The same experiments are conducted with anomalies injected at anomaly intensity of 0.05, which means 250 000 anomalous flows per 5 minutes. These flows will account for 25% to 30% of all traffic for the respective 5 minute range where the anomaly was injected. In the same manner as for anomaly intensity 0.03, we start by conducting experiments for this scenario without migration, and then with migrating different amounts of traffic, figure 5.8. The results show 100% TPR for all dimensionalities except for $k = 1$, which gives 70% TPR for 2% FPR for threshold 1.2. In the rest of the experiments the results for more than one dimensionality don't change. This is due to the high intensity of the injected anomalies. If we assume that all anomalies will be of this size, and we use more than one dimension for the final anomaly detection, then migration does not affect the results at all even when almost 40% of the traffic is moved to another location. Discussing the changes for the results for one dimension will be analogical to discussing the results for a lower anomaly intensity, as was done in the previous paragraph, because using more dimensions is done to make the detection system more sensitive to different anomaly features. Lower anomaly intensity also makes the system more sensitive because smaller anomalies can

hide better in bursty traffic compared to larger anomalies.

5.2.2 Vertical Port Scan

We continue in a similar manner by observing vertical port scans. Figure 5.10 shows the results of anomaly detection at intensity 0.01 without migration involved and with different amounts of migration. For the baseline traffic, for $k = 1$, TPR is 37%, and for $k = 2$ TPR is 87%. For the rest of the dimensionalities, it is 100%. Observing figure 5.10(b), we can see the effect of migration of 2.5% of overall traffic on the anomaly detection results. The effect is negligible because the ROC curves have only slightly shifted and TPR values have not been changed. For the same threshold without and with migration, the difference is 0.1% in the FPR. In the next experiment, figure 5.10(c), the FPR for the same threshold has increased, but at the same time the TPR has increased as well to 90%. Thus migration does not change the results significantly. Next, we increase the percentage of migrated traffic even more to 39%, figure 5.10(d). Here we can finally see significant change in the results. FPR has decreased and more dimensionalities are affected by the altered traffic. Thus, we can conclude that if more than 21% traffic is migrated, this change in traffic will be detected as an anomaly by the detection system.



(a) Without migration

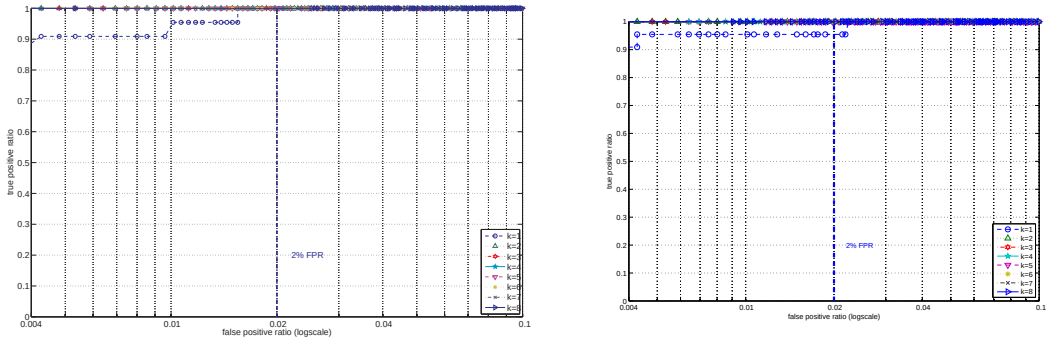
(b) 2.5% migration

(c) 21% migration

(d) 39% migration

Figure 5.10: Anomaly detection results for *alltcp* dataset for vertical port scan attacks with intensity 0.01

We have found the limit for migration size for anomaly intensity 0.01. Next, we will observe results for a higher intensity - 0.03. Figure 5.11 compares baseline traffic to traffic with 39% migration and vertical port scan attacks. The only difference between the two sets of results is the change of TPR for dimensionality 1 from 100% to 95%. Considering that no other dimensionalities have been affected, we can conclude that migration has no affect on traffic with high intensity of vertical port scan attacks. Most probably the feature that is analyzed by the anomaly detection tool is the flow count because it has been reduced almost 40%. Due to the high intensity of anomalies and the nature of vertical port scan anomalies, migration is considered as normal traffic.



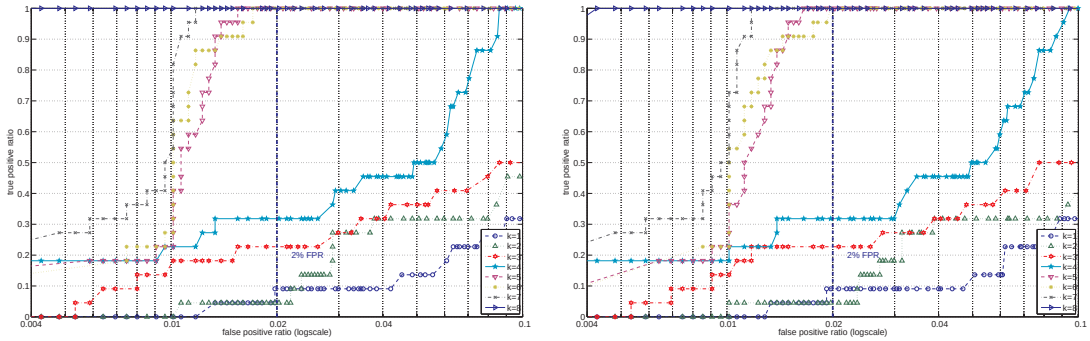
(a) Without migration (b) 39% migration

Figure 5.11: Anomaly detection results for *alltcp* dataset for vertical port scan attacks with intensity 0.03

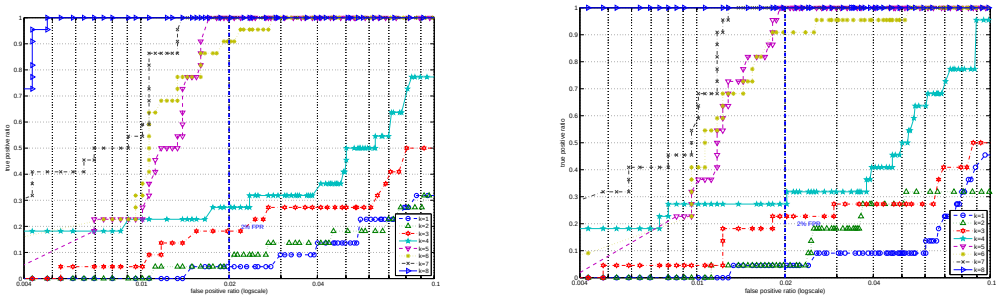
5.2.3 Horizontal Port Scan

Judging by results for the *mixed* dataset, horizontal port scans are more sensitive to the anomaly detection tool used and to inserting migration than vertical port scans. Hence, we expect to see a higher impact on anomaly detection accuracy by migration. For anomaly intensity 0.01, the baseline results by the anomaly detection tool for TPR are below 30% for the first 4 dimensionalities and 100% for the other dimensionalities. These results are visible in figure 5.12(a). Hence, the system is very sensitive to this intensity compard to the results we got with vertical port scans. However, the migration of 2.5% of all traffic still has no effect on the anomaly detection results. There is a slight shift in the ROC curves, but it is so small that even at this intensity there is no change in the final results. However, if we look closer, we can notice that the threshold which gives the original TPR and FPR results shifts to the right, and the FPR increases by 0.5% for dimensionality 4. This is still a very small change, and, besides, disregarding the threshold value, the final rates with and without migration are the same. Hence, we still consider this migration as having no affect on results. The next scenario we look at is migrating 10% of overall traffic, figure 5.12(c). Here we can already see that migration affects the accuracy of anomaly detection. The TPR has decreased for most of the dimensionalities. This means that the migrated traffic held features that were significant for defining the baseline traffic. When the flow and byte counts decrease and the distribution of the IP address and port changes, the detection tool assigns anomaly scores that exceed the previously used threshold. At this stage, we establish that migrating more than 10% of the traffic will influence the anomaly detection by decreasing the FPR and TPR values. We confirm this with one more experiment for migrating 21% of overall traffic, presented in figure 5.12(d). Results look similar to the previos migration experiment, but results for each threshold are actually shifted to the right.

As with every other anomaly type, we also conduct the same experiments with a higher intensity for injected anomalies. The baseline traffic detection results are shown in figure 5.13. Only the first two dimensions do not give a result of 100% TPR. The first dimensionality has 37% true positives and the second - 60%. By increasing the size of migrated traffic we want to observe how these values change. We start by 10% migration because this was the determined limit for the lower anomaly intensities for the same anomaly type. In this scenario we can observe that for $k = 1$ the threshold value producing the baseline rates has moved to the right. The TPR value stays the same, and the FPR has increase to 3%. Observing the results for $k = 2$ are more interesting. At first, it appears as if the results have improved because the TPR value for 2% FPR has increased with 10%. However, that value does not correspond to the same threshold. The point corresponding to the original 2% FPR has increased only slightly to 2.1% FPR, and TPR has increased to 70%. Hence, the results really have improved for this dimensionality. Since all the other dimensions already give 100% TPR, we can conclude that migrating 10% of the traffic actually improved the horizontal scan detection. Removing part of the traffic could have removed some extra noise or the nature of migration traffic features and horizontal port scan traffic features can affect each other. For example, it is possible that removing some services reduces the range of source IPs for incoming traffic, and horizontal port scans change the distribution of destination IPs by adding more counts towards some of the IPs, thus both source



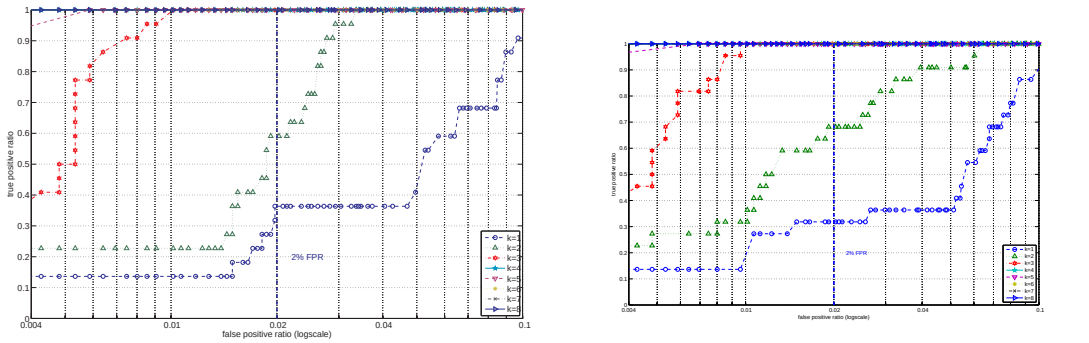
(a) Without migration (b) 2.5% migration



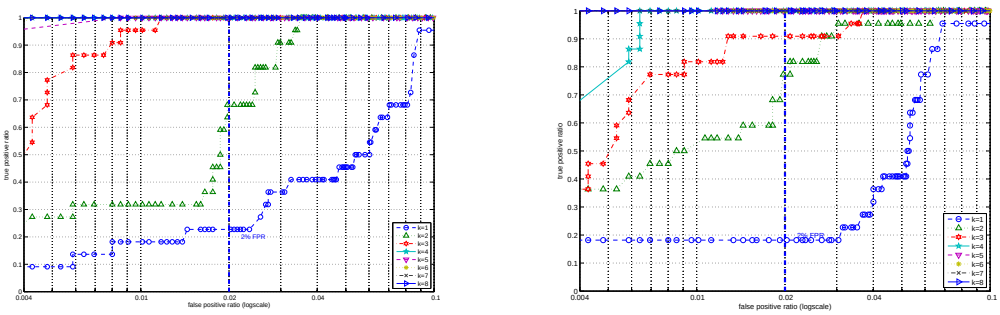
(c) 10% migration (d) 21% migration

Figure 5.12: Anomaly detection results for *alltcp* dataset for horizontal port scan attacks with intensity 0.01

and destination IP distributions are changed, but in an opposite way. When migration occurs, the change resulting from a horizontal port scan is more vivid.



(a) Without migration (b) 10% migration



(c) 21% migration (d) 39% migration

Figure 5.13: Anomaly detection results for *alltcp* dataset for horizontal port scan attacks with intensity 0.03

5.3 Summary

To summarize the findings of all experiments, anomaly type, anomaly intensity and migration size greatly affect the results produced by the anomaly detection tool. Migration with DDoS affects final results more than migration with port scans. In addition, at high anomaly intensities for any type of anomaly, migration does not show a significant influence on detection results when multiple dimensions are used. Using more dimensions usually gives better detection results in the cases we have shown. However, more dimensions means that more noise is included in the traffic trace. If the anomalies are very small and many dimensions are used, then they will hide with the bursty traffic and will not be detected. That is why we do not focus on the highest possible dimensionality, but just try to observe the effect on all of them, because the goal of this project is not finding the best anomaly detection method, but to observe what changes migration causes.

Tables A.4, A.5, A.6 in Appendix A include the list of all experiments from the experiments tables and the results for each experiment for each dimension k. The value of the dimension is taken for 2% False Positive Rate (FPR) and is the True Positive Rate (TPR). The results show how this rate changes with migration for different anomaly types and intensities or amount of traffic migrated.

Chapter 6

Conclusion

This chapter sums up the conclusions we have made based on background research and experiments. It provides the answer to the project goal we set at the beginning of this research: if and how virtual service migration affects the performance of anomaly detection. We continue by listing the key factors that contribute to the final answers and we also present how this work can be extended in the future.

6.1 Summary

This work started by analyzing existing research about data center archicture and traffic. Based on state-of-the-art models, we decided on the typical cloud infrastructure, namely a 3-tier archicture, consisting of core, aggregation and ToR switches. We also established that internal traffic can be very complex and thus, we focused on external cloud traffic. Using SWITCH incoming and outgoing traffic, 200 services were selected to represent cloud services in our model. Next, we researched anomaly detection and decided that traffic analysis needs to be done at the edge of the architecture, namely the core switches. We continue by analyzing the traffic from all of the said switches together in order to also catch anomalies whose destinations are spread throughout the data center. Since the goal of this work is related to migration's affect on traffic, we also researched how virtual service migration is done within the data center or wide-area. The reasons for migration such as load balancing, maintenance, and power failure are also an important part of the migration modeling. Once the system description was established, we continued by constructing methodology steps to process traffic data and analyze it for injected anomalies. By extending an existing flow-reading framework, we filtered SWITCH data for the selected services, split it in 5-minute-bins and stored it. We also generated anomalies in the same format. Using all this data, various scenarios were created by constructing timeseries based on a subset of services and anomalies. In addition, migration was incorporated into constructing these timeseries, which allowed for simulating virtual service migration of various sizes. Finally, we ran a number of experiments to determine how virtual service migration affects anomaly detection.

6.2 Key Findings

Based on the conducted experiments we can conclude that the effect of virtual service migration on anomaly detection performance depends on several factors. First, it depends on the type of anomaly that has occured. We have experienced different results for DDoS attacks and port scans. For the same intensity of injecting anomalies and for the same amount of traffic migrated DDoS performs worse than port scans. It gives a lower True Positive Rate and migration contributes to the decrease of this value. In addition, we can differentiate the results from the two types of port scans - vertical and horizontal. The anomaly detection tool detected anomalies with higher accuracy when they were vertical port scans as opposed to when the anomalies were horizontal port scans. This is due to the nature of the attacks and the anomaly detection tool specifications. However, this accuracy affects migration as well. The better the anomaly detection performs without migration, the less difference migration makes. When the system is already very sensitive to attacks or bursty traffic, it increases its false positives by

adding migration to them. For example, for the port scans experiments on the *mixed* dataset even when 30% of the traffic was migrated, the effect on the anomaly detection was not as huge as the one with DDoS anomalies.

Another factor on the effect of virtual service migration on anomaly detection is the anomaly intensity. We have observed that the higher the anomaly intensity is, the lower the effect of migration on the anomaly detection system. When anomalies are much stronger than the migration, the latter remains hidden for the analysis tool and has no consequence for the final results. When anomalies are very small in size, it is harder for the anomaly detection tool to identify them due to a lot of bursty traffic around, which presents itself similar to the anomalies. In this case, migration wants to be considered as part of the normal traffic, hence, it is just like the bursty traffic - it shows changes to the trace, but is not an anomaly. However, when the detection tool is sensitive due to lower sized anomalies, part of the bursty traffic it classifies as anomalous is the migration.

The size of the anomalies is not the only factor related to traffic size that changes the anomaly detection results. The size of the migrated traffic as part of the total traffic also affects the final results. The lower the percentage of migrated traffic is, the less chance it can be detected as an anomaly. It has been established that for up to 10% the migration has no significant effect on the anomaly detection results. This percentage varies depending on the type of anomaly and the anomaly intensity. Consequently, we can say that the combination of all of these three factors (anomaly type, anomaly intensity, and size of migration) determines the rate at which virtual service migration affects anomaly detection analysis.

To sum up, Table 6.1 shows the final quantitative results on the effect of virtual service migration on anomaly detection. For each anomaly type and anomaly intensity combination, a tested limit for percentage of traffic that can be migrated without significant consequences on the anomaly detection results is given. Intensity is presented by a percentage of anomalous traffic out of total traffic at the time of injection.

Type of Migration	Anomaly Intensity	Limit of Migration Size
DDoS	15%	10%
DDoS	25%	39%
Horizontal port scan	5%	10%
Horizontal port scan	15%	21%
Vertical port scan	5%	21%
Vertical port scan	15%	39%

Table 6.1: Conclusions on migration impact on anomaly detection

6.3 Future work

The goal of this work was to find out what effect virtual service migration has on existing anomaly detection techniques. The focus of this work was not to find the best anomaly detection system but to use an existing one based on state-of-the-art analysis techniques for anomaly detection. Hence, one way of expanding on this project is to apply the same system and experiments to a different anomaly detection technique to observe if better detection will classify migration as normal traffic or give worse results by detecting small changes in traffic such as migration.

Another direction of extending this project is to look into local-area migration. Due to the complexity of internal traffic and the lack of any samples of internal traffic, this work focused on analyzing external traffic and wide-area migration. However, it will be interesting to find out how

different the effect of local-area migration can be on the anomaly detection system, compared to the effect of wide-area migration, analyzed in this work. In addition, to improve accuracy of results, real cloud traffic data could be used if available.

This project has established that at a certain amount of traffic migration the results of anomaly detection are affected. However, this work does not offer a solution of how to extend this migration size limit and this could be an interesting future research. Perhaps the analyzed traffic features can be extended to include more information or certain traffic features that help classify migration as an anomaly can be omitted. We have established that migration is the least detected with vertical port scan anomalies. Future work can analyze what features of migration or of vertical port scans cause this result. These conclusions can subsequently be used to improve the results for other types of attacks. Finally, this paper encourages more research into training models to recognize migration as normal traffic.

Bibliography

[1] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74, Aug. 2008.

[2] Amazon EC2 public IP ranges. https://forums.aws.amazon.com/ann.jspa?annID=1701, October 2012.

[3] Anti-spam Spamhaus up again after 75Gbps Distributed Denial of Service (DDoS) attacks. http://www.ddosattacks.net/2013/03/21/anti-spam-spamhaus-up-again-after-75gbps-distributed-denial-of-service-ddos-attacks/, March 2013.

[4] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Towards predictable datacenter networks. *SIGCOMM Comput. Commun. Rev.*, 41(4):242–253, Aug. 2011.

[5] P. Barford and D. Plonka. Characteristics of network traffic flow anomalies. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, IMW '01, pages 69–73, New York, NY, USA, 2001. ACM.

[6] T. Benson, A. Akella, and D. A. Maltz. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, IMC '10, pages 267–280, New York, NY, USA, 2010. ACM.

[7] T. Benson, A. Anand, A. Akella, and M. Zhang. Understanding data center traffic characteristics. In *Proceedings of the 1st ACM workshop on Research on enterprise networking*, WREN '09, pages 65–72, New York, NY, USA, 2009. ACM.

[8] S. Berger, R. Cáceres, K. Goldman, D. Pendarakis, R. Perez, J. R. Rao, E. Rom, R. Sailer, W. Schildhauer, D. Srinivasan, S. Tal, and E. Valdez. Security for the cloud infrastructure: Trusted virtual data center implementation. *IBM J. Res. Dev.*, 53(4):560–571, July 2009.

[9] B. Betts. Leave nothing behind when migrating virtual machines. http://www.theregister.co.uk/2011/09/28/virtual_machines/, September 2011.

[10] M. Bienkowski, A. Feldmann, D. Jurca, W. Kellerer, G. Schaffrath, S. Schmid, and J. Widmer. Competitive analysis for service migration in VNets. In *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, VISA '10, pages 17–24, New York, NY, USA, 2010. ACM.

[11] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg. Live wide-area migration of virtual machines including local persistent state. In *Proceedings of the 3rd international conference on Virtual execution environments*, VEE '07, pages 169–179, New York, NY, USA, 2007. ACM.

[12] D. Brauckhoff, K. Salamatian, and M. May. A signal processing view on packet sampling and anomaly detection. In *Proceedings of the 29th conference on Information communications*, INFOCOM'10, pages 713–721, Piscataway, NJ, USA, 2010. IEEE Press.

[13] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, and A. Lakhina. Impact of packet sampling on anomaly detection metrics. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, pages 159–164, New York, NY, USA, 2006. ACM.

[14] A. Chandak, K. Jaju, A. Kanfade, P. Lohiya, and A. Joshi. Dynamic load balancing of virtual machines using QEMU-KVM. *International Journal of Computer Applications*, 46(6), May 2012.

[15] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[16] C.-W. Chang, A. Gerber, B. Lin, S. Sen, and O. Spatscheck. Network DVR: a programmable framework for application-aware trace collection. In *Proceedings of the 11th international conference on Passive and active measurement*, PAM'10, pages 191–200, Berlin, Heidelberg, 2010. Springer-Verlag.

[17] Cisco IOS NetFlow version 9 flow-record format. Cisco White Paper, May 2011.

[18] DDoS attack strikes american express site. http://www.ddosattacks.net/2013/03/30/ddos-attack-strikes-american-express-site/, March 2013.

[19] Distributed denial of service-DDoS: 6 banks hit on same day. http://www.ddosattacks.net/2013/03/19/distributed-denial-of-service-ddos-6-banks-hit-on-same-day/, March 2013.

[20] J. Dix. Who has responsibility for cloud security? http://www.cio.com/article/730729, March 2013.

[21] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat. Helios: a hybrid electrical/optical switch architecture for modular data centers. *SIGCOMM Comput. Commun. Rev.*, 41(4):339–350, Aug. 2010.

[22] A. Fischer, A. Fessi, G. Carle, and H. de Meer. Wide-area virtual machine migration as resilience mechanism. In *Proceedings of the 2011 IEEE 30th Symposium on Reliable Distributed Systems Workshops*, SRDSW '11, pages 72–77, Washington, DC, USA, 2011. IEEE Computer Society.

[23] Flame. http://flame.ee.ethz.ch, March 2013.

[24] Flowbox. https://github.com/FlowBox, March 2013.

[25] P. Gill, N. Jain, and N. Nagappan. Understanding network failures in data centers: measurement, analysis, and implications. *SIGCOMM Comput. Commun. Rev.*, 41(4):350–361, Aug. 2011.

[26] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: a scalable and flexible data center network. *Commun. ACM*, 54(3):95–104, Mar. 2011.

[27] A. Greenberg and D. A. Maltz. What goes into a data center? http://research.microsoft.com/apps/pubs/default.aspx?id=81782.

[28] S. Guha, J. Chandrashekar, N. Taft, and K. Papagiannaki. How healthy are today's enterprise networks? In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, IMC '08, pages 145–150, New York, NY, USA, 2008. ACM.

[29] B. Hayes. Cloud computing. *Commun. ACM*, 51(7):9–11, July 2008.

[30] S. Kandula, J. Padhye, and P. Bahl. Flyways to de-congest data center networks. *In Proc. of HotNets*, 2005.

[31] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken. The nature of data center traffic: measurements & analysis. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, pages 202–208, New York, NY, USA, 2009. ACM.

[32] A. Lakhina, M. Crovella, and C. Diot. Characterization of network-wide anomalies in traffic flows. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, IMC '04, pages 201–206, New York, NY, USA, 2004. ACM.

[33] A. Li, L. Gu, and K. Xu. Fast anomaly detection in large data centers. *Proc. of the 2010 IEEE Global Commuincations Conference*, 2010.

[34] A. A. Mahimkar, H. H. Song, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and J. Emmons. Detecting the performance impact of upgrades in large operational networks. *SIGCOMM Comput. Commun. Rev.*, 41(4):303–314, Aug. 2010.

[35] J. Mai, A. Sridharan, C.-N. Chuah, H. Zang, and T. Ye. Impact of packet sampling on portscan detection. *IEEE J.Sel. A. Commun.*, 24(12):2285–2298, Dec. 2006.

[36] E. Messmer. Cloud security to be most disruptive technology of 2013. http://www.networkworld.com/news/2013/010313-cloud-security-265437.html, January 2013.

[37] E. Messmer. Growing confidence in cloud security. http://www.networkworld.com/news/2013/010213-outlook-security-265293.html, January 2013.

[38] J. Mudigonda, P. Yalagandula, J. Mogul, B. Stiekes, and Y. Pouffary. Netlord: a scalable multi-tenant network architecture for virtualized datacenters. *SIGCOMM Comput. Commun. Rev.*, 41(4):62–73, Aug. 2011.

[39] M. Nelson, B.-H. Lim, and G. Hutchins. Fast transparent migration for virtual machines. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '05, pages 25–25, Berkeley, CA, USA, 2005. USENIX Association.

[40] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney. A first look at modern enterprise traffic. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, IMC '05, pages 2–2, Berkeley, CA, USA, 2005. USENIX Association.

[41] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. Improving datacenter performance and robustness with multipath TCP. *SIGCOMM Comput. Commun. Rev.*, 41(4):266–277, Aug. 2011.

[42] F. Silveira, C. Diot, N. Taft, and R. Govindan. Astute: detecting a different class of traffic anomalies. *SIGCOMM Comput. Commun. Rev.*, 41(4):267–278, Aug. 2010.

[43] E. Silvera, G. Sharaby, D. Lorenz, and I. Shapira. IP mobility to support live migration of virtual machines across subnets. In *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference*, SYSTOR '09, pages 13:1–13:10, New York, NY, USA, 2009. ACM.

[44] SWITCH, the swiss education and research network. http://www.switch.ch, March 2013.

[45] B. Tellenbach, D. Brauckhoff, and M. May. Impact of traffic mix and packet sampling on anomaly visibility. In *Proceedings of the 2008 The Third International Conference on Internet Monitoring and Protection*, ICIMP '08, pages 31–36, Washington, DC, USA, 2008. IEEE Computer Society.

[46] B. Tellenbach, M. Burkhart, D. Schatzmann, D. Gugelmann, and D. Sornette. Accurate network anomaly classification with generalized entropy metrics. *Comput. Netw.*, 55(15):3485–3502, Oct. 2011.

[47] B. Tellenbach, M. Burkhart, D. Sornette, and T. Maillart. Beyond shannon: Characterizing internet traffic with generalized entropy metrics. In *Proceedings of the 10th International Conference on Passive and Active Network Measurement*, PAM '09, pages 239–248, Berlin, Heidelberg, 2009. Springer-Verlag.

[48] Transitioning to the private cloud with confidence. https://www.cisco.com/en/US/solutions/collateral/ns340/ns517/ns224/ns376/whitepaper_c11-714844.pdf, 2012.

[49] Twenty-one experts define cloud computing. http://www.virtualizationconference.com/node/612375/print, November 2012.

[50] B. Vamanan, J. Hasan, and T. Vijaykumar. Deadline-aware datacenter TCP (D2TCP). *SIGCOMM Comput. Commun. Rev.*, 42(4):115–126, Aug. 2012.

[51] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55, Dec. 2008.

Appendix A

Experiments and Results

Experiment ID	Timeseries	Anomaly Set	Anomaly Intensity	Migration
1	anomalous2	ddos2	0.001	no
2	migration2	ddos2	0.001	5.8%
3	migration3	ddos2	0.001	30%
4	anomalous3	ddos6	0.004	no
5	migration4	ddos6	0.004	5.8%
6	migration5	ddos6	0.004	30%
7	anomalous3	ddos6	0.004	special
15	portscan1	portscan1	0.001	no
16	portscan2	portscan2	0.004	no
17	portscan3	portscan5	0.001	no
18	portscan4	portscan6	0.004	no
23	migrps1	portscan1	0.001	5.8%
24	migrps2	portscan2	0.004	5.8%
25	migrps3	portscan5	0.001	5.8%
26	migrps4	portscan6	0.004	5.8%
27	migrps5	portscan1	0.001	30%
28	migrps6	portscan2	0.004	30%
29	migrps7	portscan5	0.001	30%
30	migrps8	portscan6	0.004	30%

Table A.1: List of experiments for *mixed* dataset

Experiment ID	Timeseries	Anomaly Set	Anomaly Intensity	Migration
8	anomalous	ddos4	0.1	no
9	anomalous2	ddos8	0.03	no
10	anomalous3	ddos9	0.05	no
11	migration1	ddos8	0.03	2.5%
12	migration2	ddos9	0.05	2.5%
13	migration3	ddos8	0.03	0.27%
14	migration4	ddos9	0.05	0.27%
39	migration5	ddos8	0.03	10%
40	migration6	ddos9	0.05	10%
45	migration7	ddos8	0.03	21%
46	migration8	ddos9	0.05	21%
51	migration9	ddos8	0.03	28%
52	migration10	ddos9	0.05	28%
53	migration11	ddos8	0.03	39%
54	migration12	ddos9	0.05	39%

Table A.2: List of DDoS experiments for *alltcp* dataset

Experiment ID	Timeseries	Anomaly Set	Anomaly Intensity	Migration
19	portscan1	portscan3	0.01	no
20	portscan2	portscan4	0.03	no
21	portscan3	portscan7	0.01	no
22	portscan4	portscan8	0.03	no
31	migrps1	portscan3	0.01	2.5%
32	migrps2	portscan4	0.03	2.5%
33	migrps3	portscan7	0.01	2.5%
34	migrps4	portscan8	0.03	2.5%
35	migrps5	portscan3	0.01	10%
36	migrps6	portscan4	0.03	10%
37	migrps7	portscan7	0.01	10%
38	migrps8	portscan8	0.03	10%
41	migrps9	portscan3	0.01	21%
42	migrps10	portscan4	0.03	21%
43	migrps11	portscan7	0.01	21%
44	migrps12	portscan8	0.03	21%
47	migrps13	portscan3	0.01	28%
48	migrps14	portscan4	0.03	28%
49	migrps15	portscan7	0.01	28%
50	migrps16	portscan8	0.03	28%
55	migrps17	portscan3	0.01	39%
56	migrps18	portscan4	0.03	39%
57	migrps19	portscan7	0.01	39%
58	migrps20	portscan8	0.03	39%

Table A.3: List of port scan experiments for *alltcp* dataset

Experiment ID	Anomaly Intensity	Migration	k=1	k=2	k=3	k =4	k=5	k=6	k=7	k=8
1	0.001	no	0%	10%	5%	5%	22%	90%	95%	100%
2	0.001	5.8%	0%	0%	0%	0%	5%	90%	90%	100%
3	0.001	30%	0%	5%	0%	0%	5%	18%	28%	41%
4	0.004	no	100%	100%	100%	100%	100%	100%	100%	100%
5	0.004	5.8%	81%	100%	95%	100%	100%	100%	100%	100%
6	0.004	30%	20%	90%	80%	80%	100%	100%	100%	100%
7	0.004	special	100%	100%	100%	100%	100%	100%	40%	32%
15	0.001	no	100%	100%	100%	100%	100%	100%	100%	100%
16	0.004	no	100%	100%	100%	100%	100%	100%	100%	100%
17	0.001	no	77%	95%	100%	100%	100%	100%	100%	100%
18	0.004	no	100%	100%	100%	100%	100%	100%	100%	100%
23	0.001	5.8%	100%	100%	100%	100%	100%	100%	100%	100%
24	0.004	5.8%	100%	100%	100%	100%	100%	100%	100%	100%
25	0.001	5.8%	72%	90%	86%	100%	100%	100%	100%	100%
26	0.004	5.8%	100%	100%	100%	100%	100%	100%	100%	100%
27	0.001	30%	64%	100%	100%	100%	100%	100%	100%	100%
28	0.004	30%	100%	100%	100%	100%	100%	100%	100%	100%
29	0.001	30%	40%	90%	72%	72%	100%	100%	95%	100%
30	0.004	30%	100%	100%	100%	100%	100%	100%	100%	100%

Table A.4: List of results from all *mixed* experiments

Experiment ID	Anomaly Intensity	Migration	k=1	k=2	k=3	k =4	k=5	k=6	k=7	k=8
8	0.1	no	100%	100%	100%	100%	100%	100%	100%	100%
9	0.03	no	20%	40%	100%	100%	100%	100%	100%	100%
13	0.03	0.27%	20%	40%	100%	100%	100%	100%	100%	100%
11	0.03	2.5%	20%	40%	100%	100%	100%	100%	100%	100%
39	0.03	10%	20%	60%	100%	90%	100%	100%	100%	100%
45	0.03	21%	30%	60%	90%	80%	100%	100%	100%	100%
51	0.03	28%	30%	70%	80%	80%	100%	100%	100%	100%
53	0.03	39%	10%	50%	70%	80%	100%	100%	100%	100%
10	0.05	no	70%	100%	100%	100%	100%	100%	100%	100%
14	0.05	0.27%	70%	100%	100%	100%	100%	100%	100%	100%
12	0.05	2.5%	70%	100%	100%	100%	100%	100%	100%	100%
40	0.05	10%	60%	100%	100%	100%	100%	100%	100%	100%
46	0.05	21%	60%	100%	100%	100%	100%	100%	100%	100%
52	0.05	28%	55%	100%	100%	100%	100%	100%	100%	100%
54	0.05	39%	50%	100%	100%	100%	100%	100%	100%	100%

Table A.5: List of results from all *alltcp* DDoS experiments

Experiment ID	Type of port scan	Anomaly Intensity	Migration	k=1	k=2	k=3	k =4	k=5	k=6	k=7	k=8
19	vertical	0.01	no	37%	87%	100%	100%	100%	100%	100%	100%
31	vertical	0.01	2.5%	37%	87%	100%	100%	100%	100%	100%	100%
35	vertical	0.01	10%	40%	90%	100%	100%	100%	100%	100%	100%
41	vertical	0.01	21%	37%	90%	100%	100%	100%	100%	100%	100%
47	vertical	0.01	28%	32%	87%	91%	100%	100%	100%	100%	100%
55	vertical	0.01	39%	22%	87%	87%	100%	100%	100%	100%	100%
20	vertical	0.03	no	100%	100%	100%	100%	100%	100%	100%	100%
32	vertical	0.03	2.5%	100%	100%	100%	100%	100%	100%	100%	100%
36	vertical	0.03	10%	100%	100%	100%	100%	100%	100%	100%	100%
42	vertical	0.03	21%	100%	100%	100%	100%	100%	100%	100%	100%
48	vertical	0.03	28%	97%	100%	100%	100%	100%	100%	100%	100%
56	vertical	0.03	39%	95%	100%	100%	100%	100%	100%	100%	100%
21	horizontal	0.01	no	10%	5%	23%	32%	100%	100%	100%	100%
33	horizontal	0.01	2.5%	10%	5%	23%	32%	100%	100%	100%	100%
37	horizontal	0.01	10%	5%	5%	19%	28%	100%	90%	100%	100%
43	horizontal	0.01	21%	5%	5%	22%	28%	100%	90%	100%	100%
49	horizontal	0.01	28%	0%	10%	5%	30%	50%	70%	100%	100%
57	horizontal	0.01	39%	0%	0%	5%	31%	40%	55%	90%	100%
22	horizontal	0.03	no	37%	60%	100%	100%	100%	100%	100%	100%
34	horizontal	0.03	2.5%	37%	60%	100%	100%	100%	100%	100%	100%
38	horizontal	0.03	10%	30%	70%	100%	100%	100%	100%	100%	100%
44	horizontal	0.03	21%	22%	69%	100%	100%	100%	100%	100%	100%
50	horizontal	0.03	28%	19%	69%	90%	100%	100%	100%	100%	100%
58	horizontal	0.03	39%	19%	78%	90%	100%	100%	100%	100%	100%

Table A.6: List of results from all *alltcp* port scan experiments

Appendix B

Plots

Here are placed plots of each experiment conducted during this work. The figures show ROC curves for 8 dimensionalities according to computed anomaly scores, which are then compared to predefined thresholds.

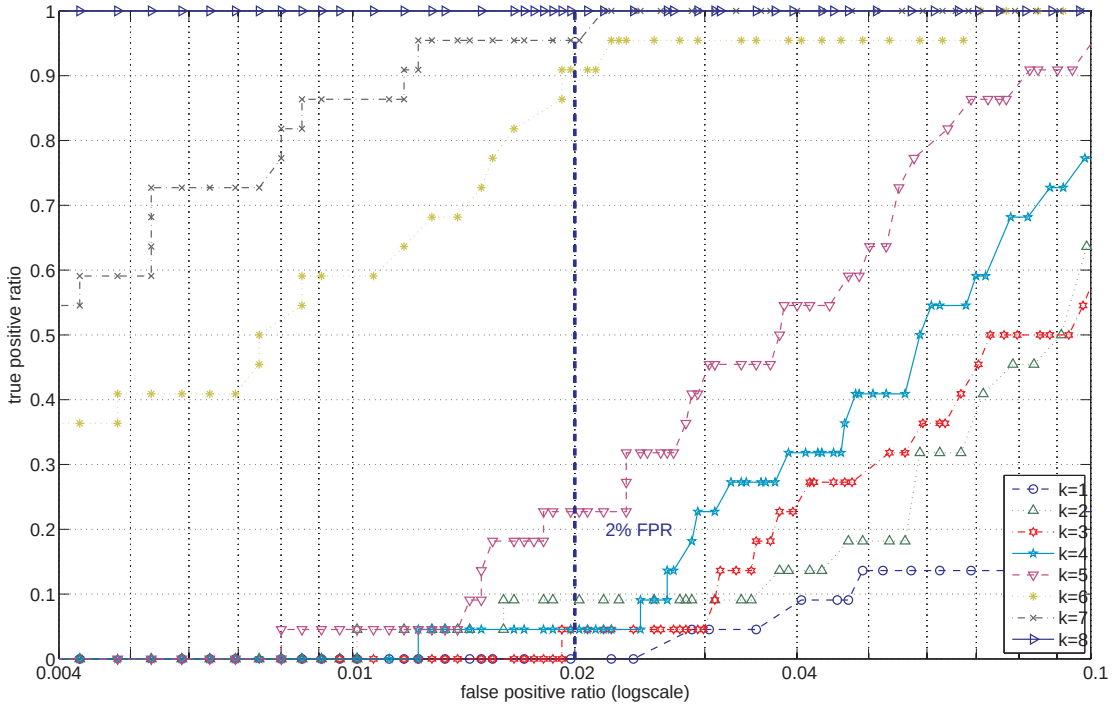


Figure B.1: *mixed*: ROC curves for DDoS with intensity 0.001 without migration

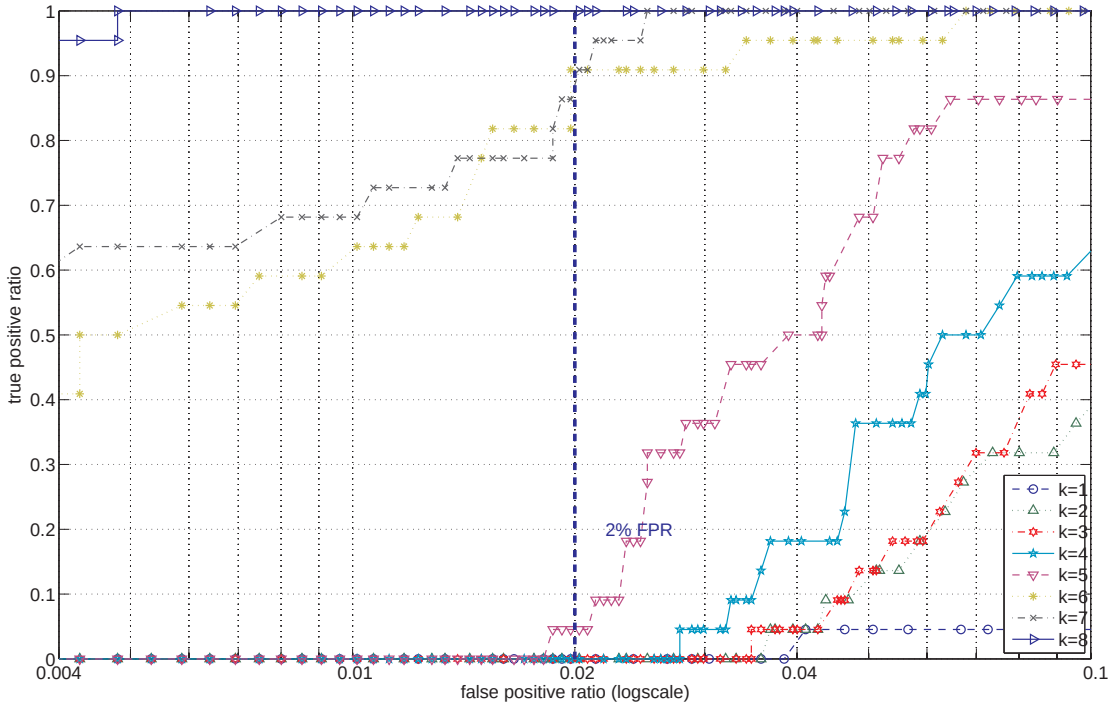


Figure B.2: *mixed*: ROC curves for DDoS with intensity 0.001 with 5.8% migration

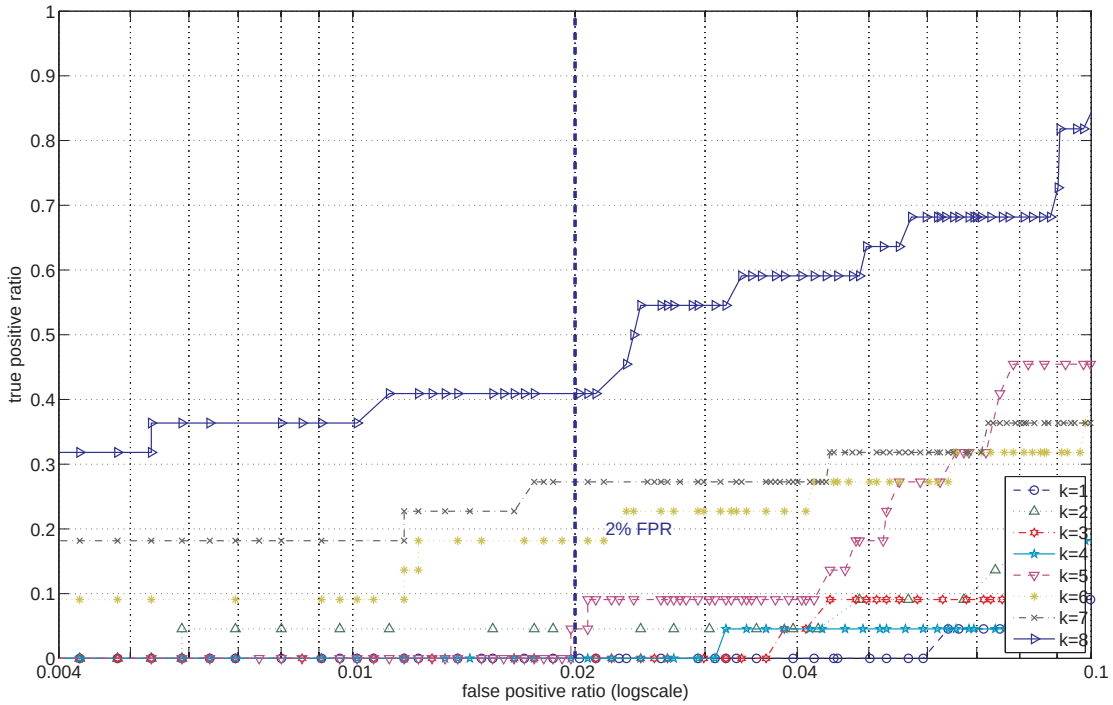


Figure B.3: *mixed*: ROC curves for DDoS with intensity 0.001 with 30% migration

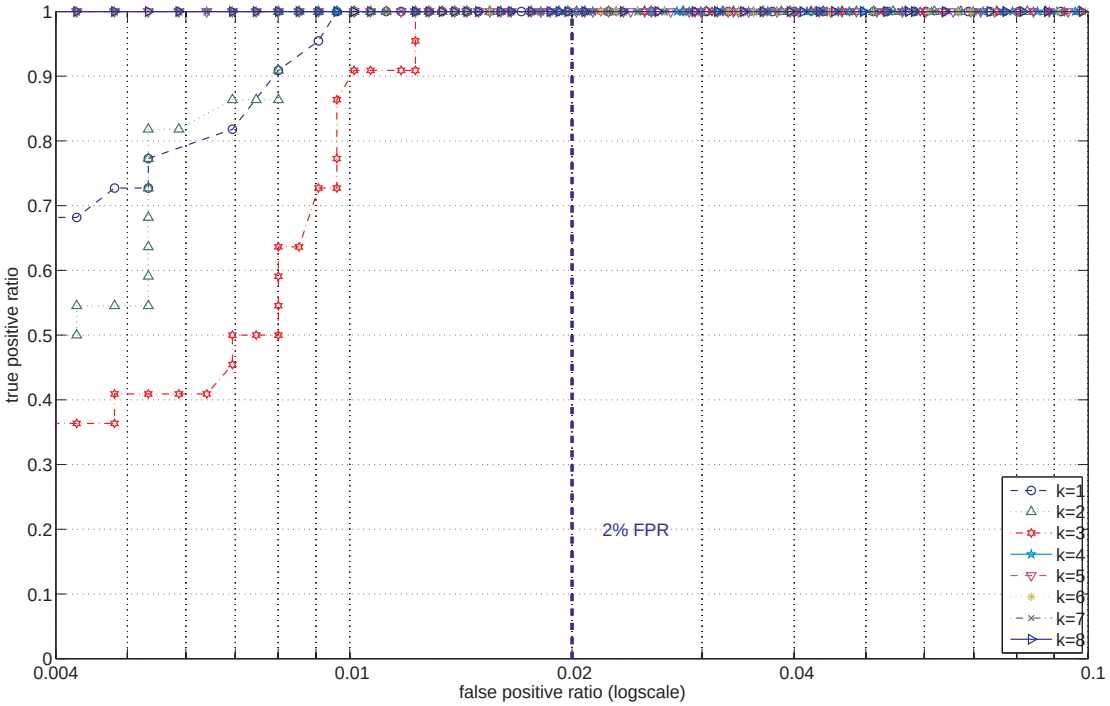


Figure B.4: *mixed*: ROC curves for DDoS with intensity 0.004 without migration

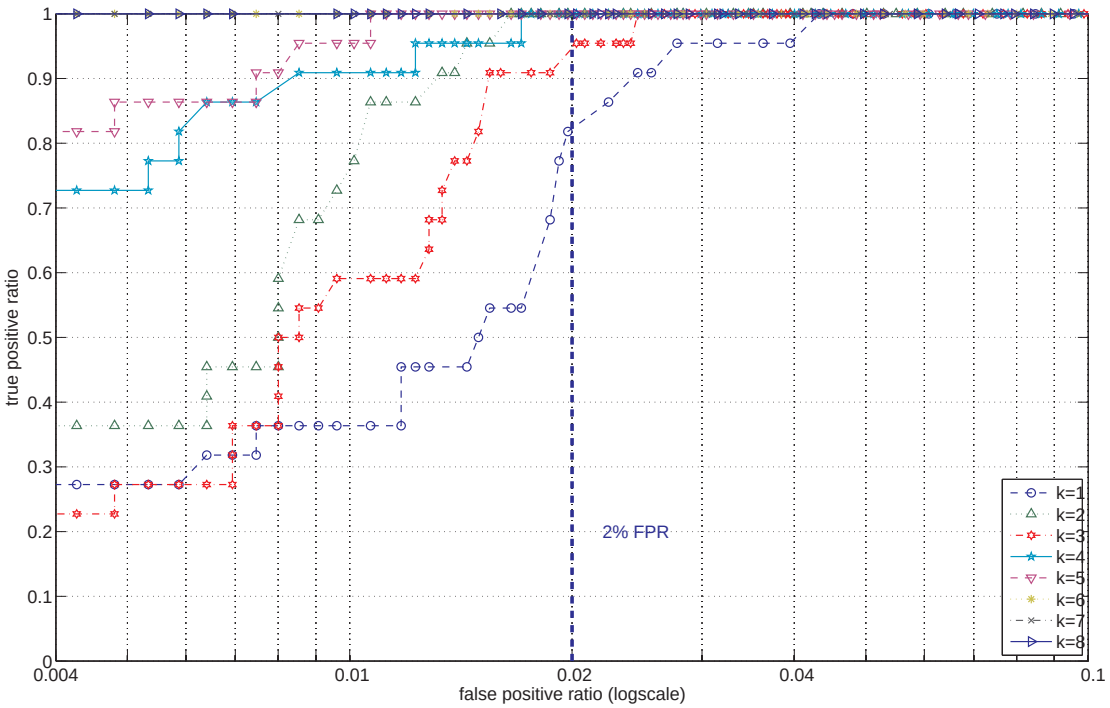


Figure B.5: *mixed*: ROC curves for DDoS with intensity 0.004 with 5.8% migration

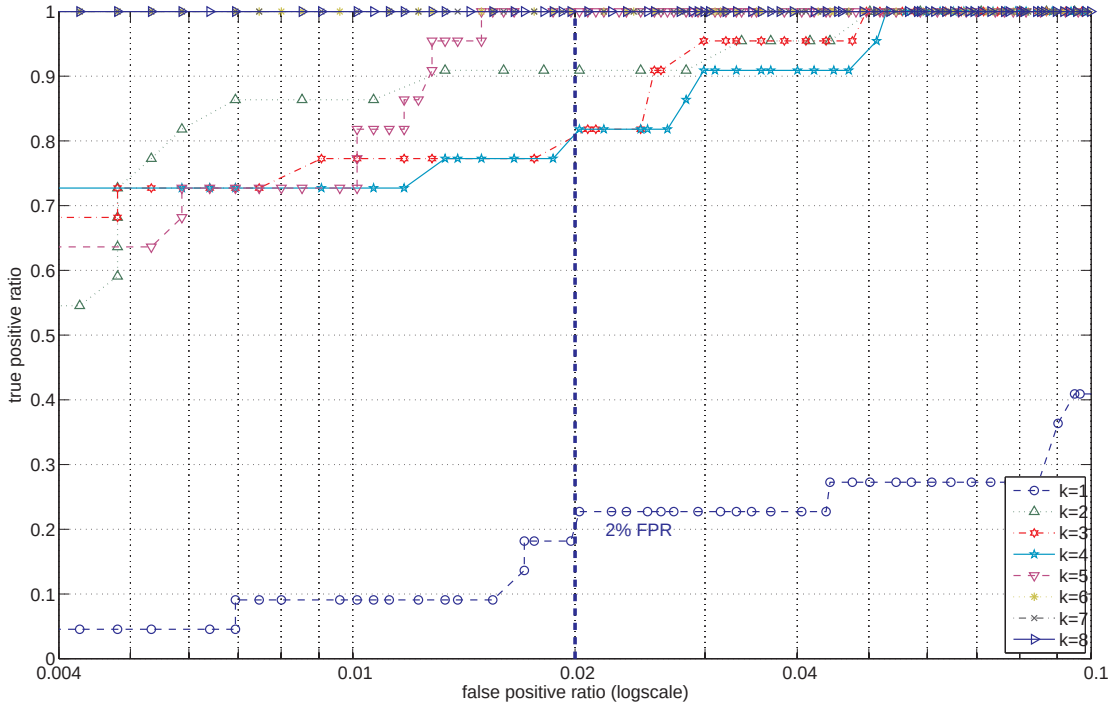


Figure B.6: *mixed*: ROC curves for DDoS with intensity 0.001 with 30% migration

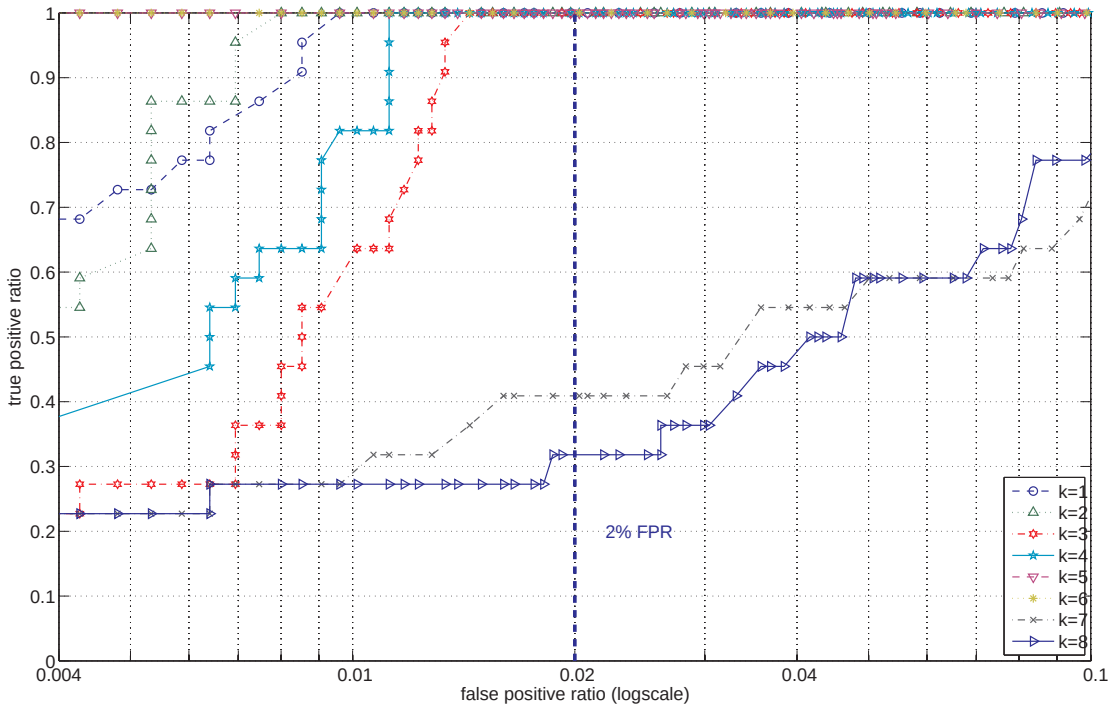


Figure B.7: *mixed*: ROC curves for DDoS for training with migration

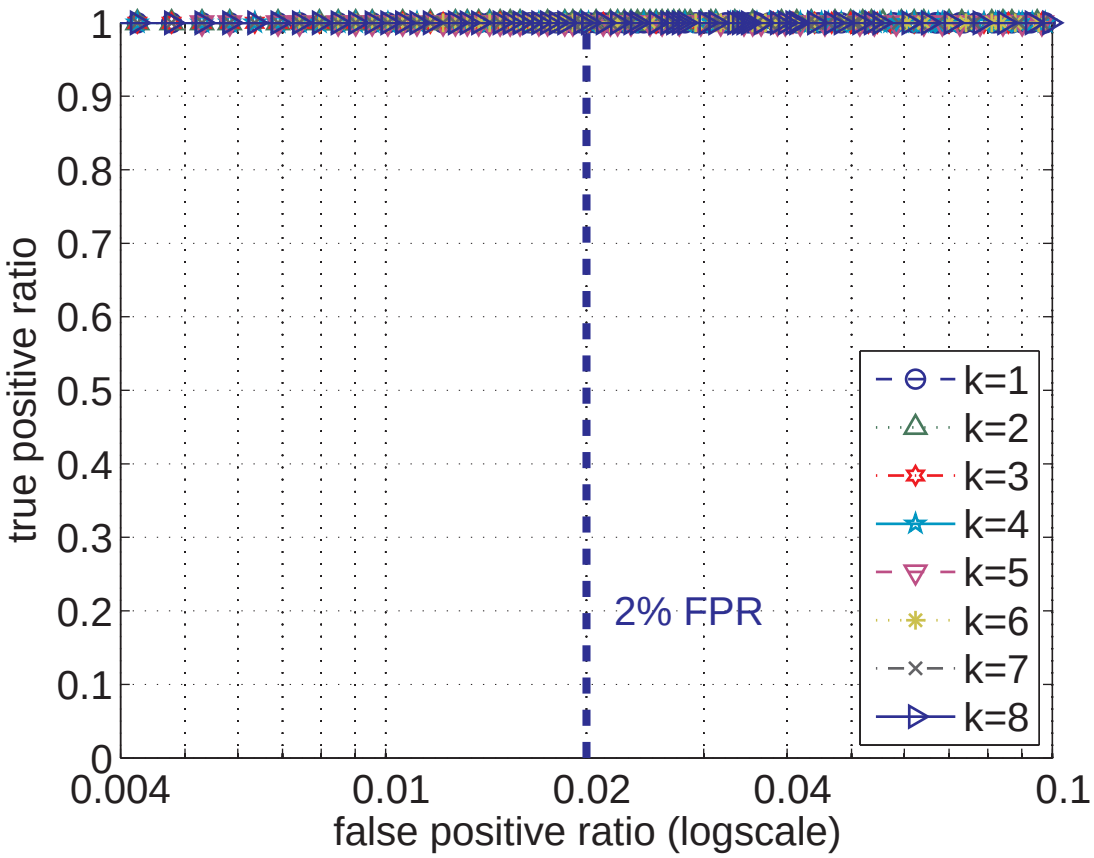


Figure B.8: *alltcp*: ROC curves for DDoS with intensity 0.1 without migration

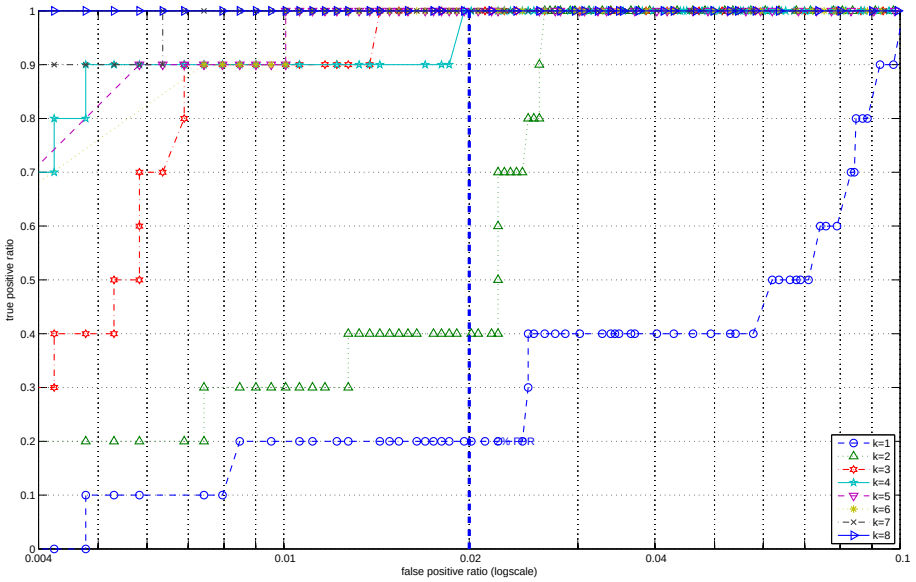


Figure B.9: *alltcp*: ROC curves for DDoS with intensity 0.03 without migration

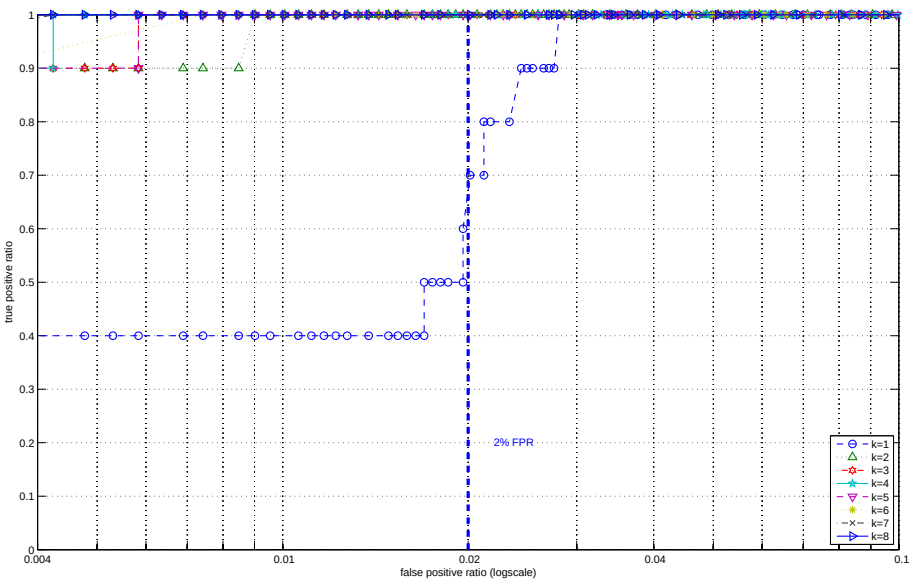


Figure B.10: *alltcp*: ROC curves for DDoS with intensity 0.05 without migration

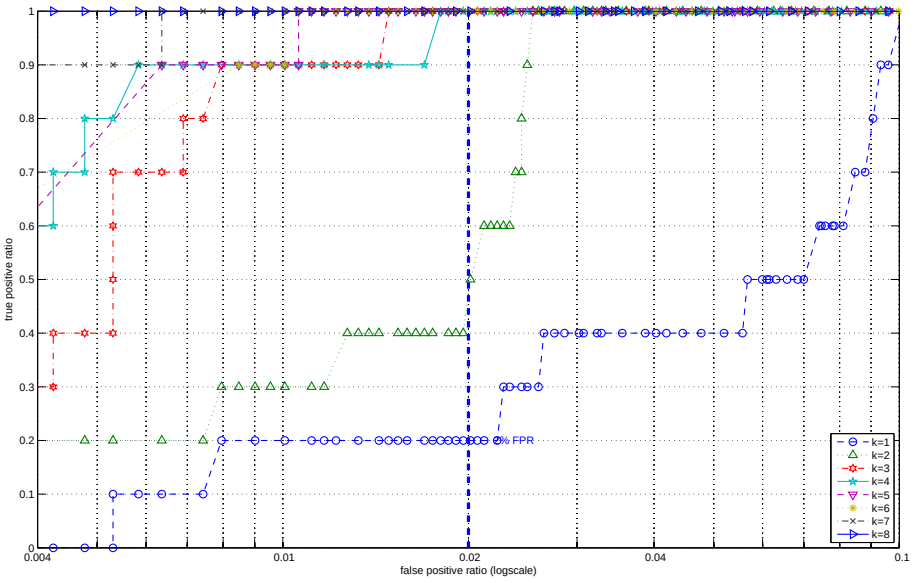


Figure B.11: *alltcp*: ROC curves for DDoS with intensity 0.03 with 2.5% migration

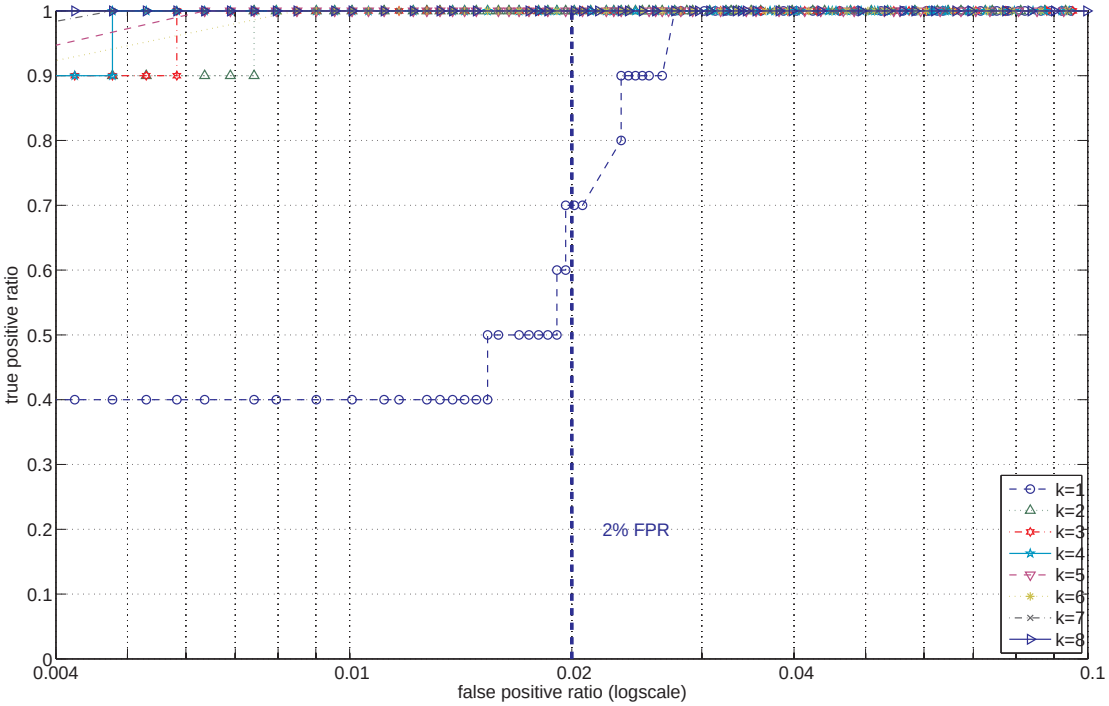


Figure B.12: *alltcp*: ROC curves for DDoS with intensity 0.05 with 2.5% migration

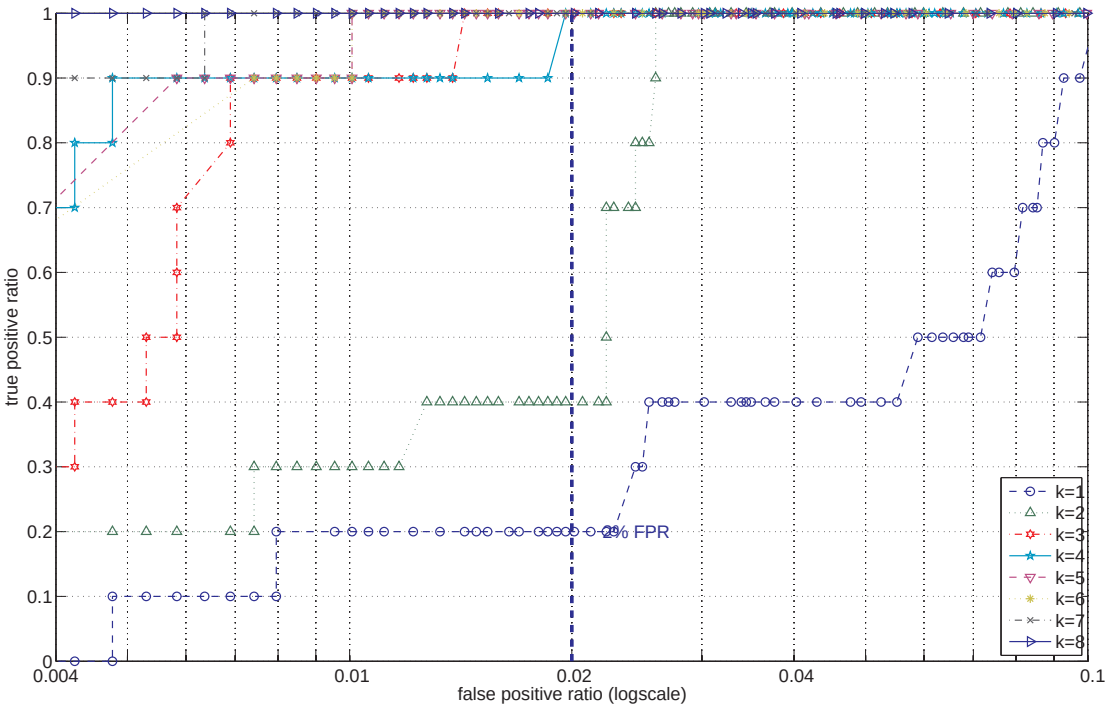


Figure B.13: *alltcp*: ROC curves for DDoS with intensity 0.03 with 0.27% migration

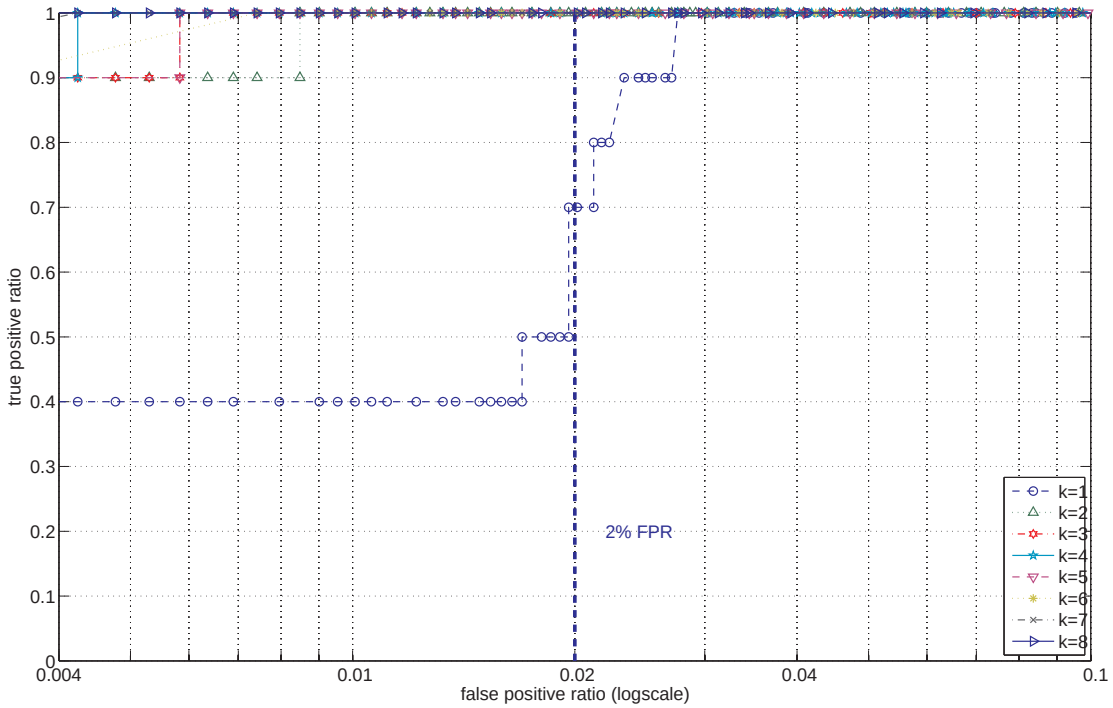


Figure B.14: *alltcp*: ROC curves for DDoS with intensity 0.05 with 0.27% migration

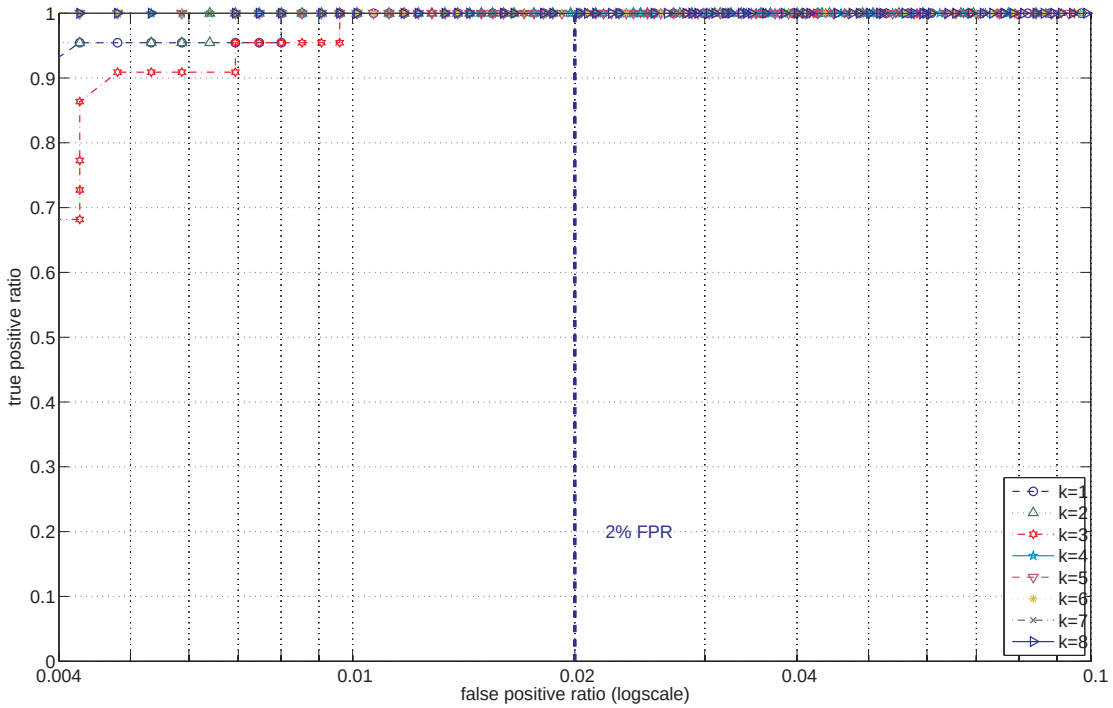


Figure B.15: *mixed*: ROC curves for vertical port scan with intensity 0.001 without migration

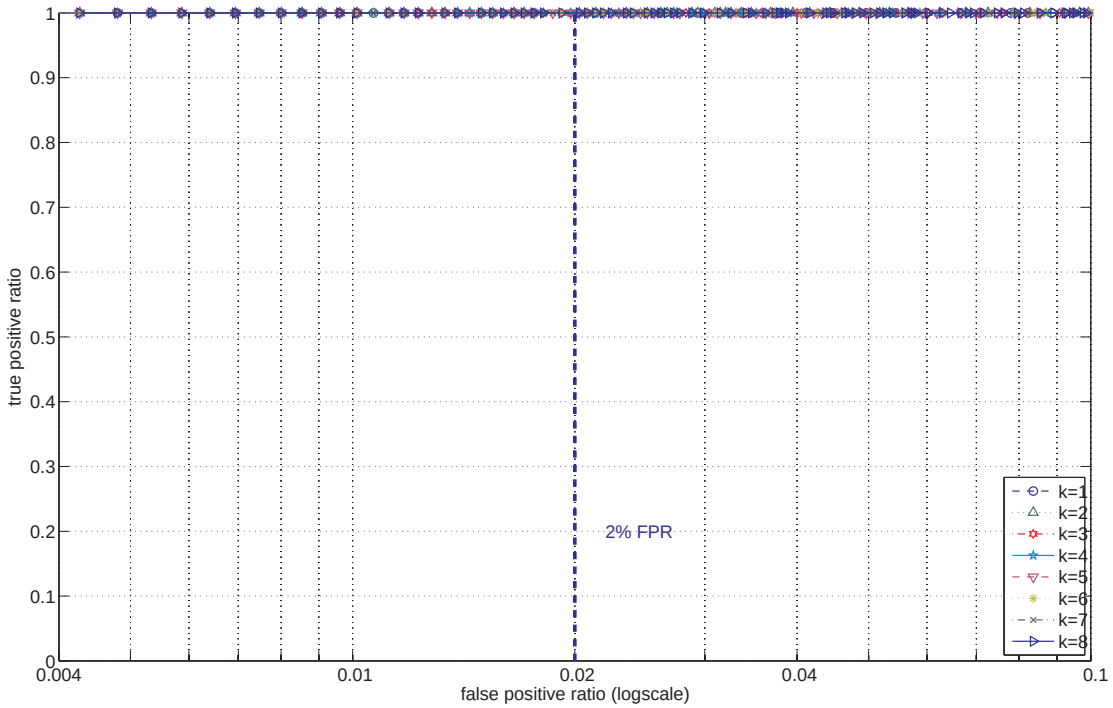


Figure B.16: *mixed*: ROC curves for vertical port scan with intensity 0.004 without migration

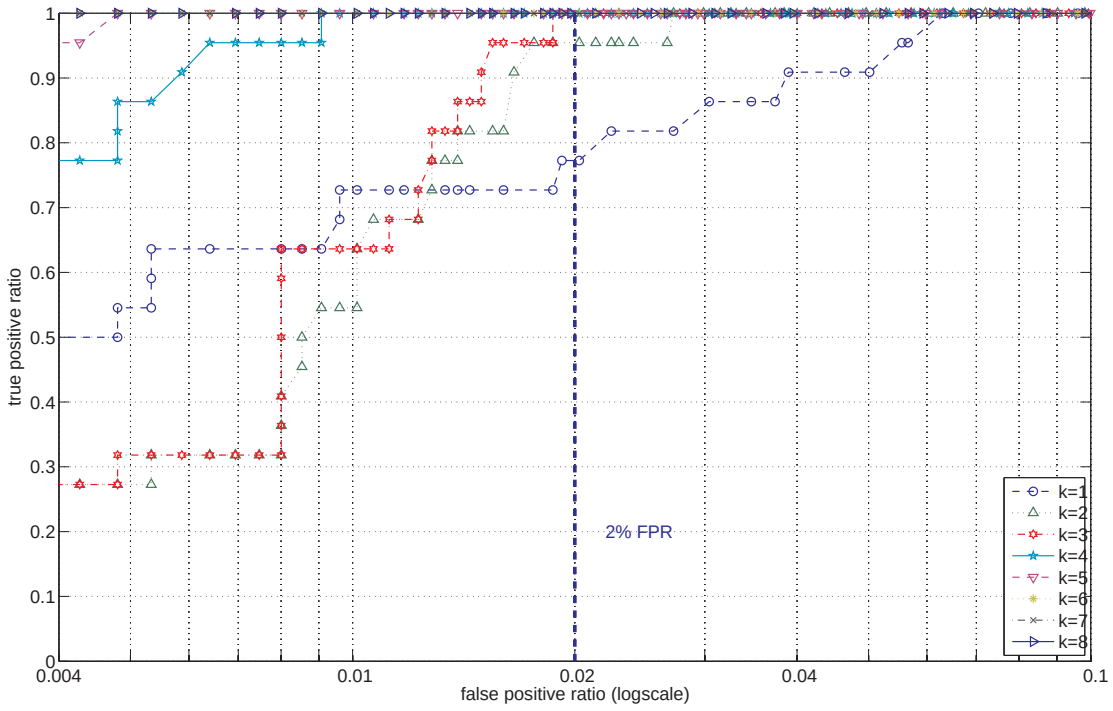


Figure B.17: *mixed*: ROC curves for horizontal port scan with intensity 0.001 without migration

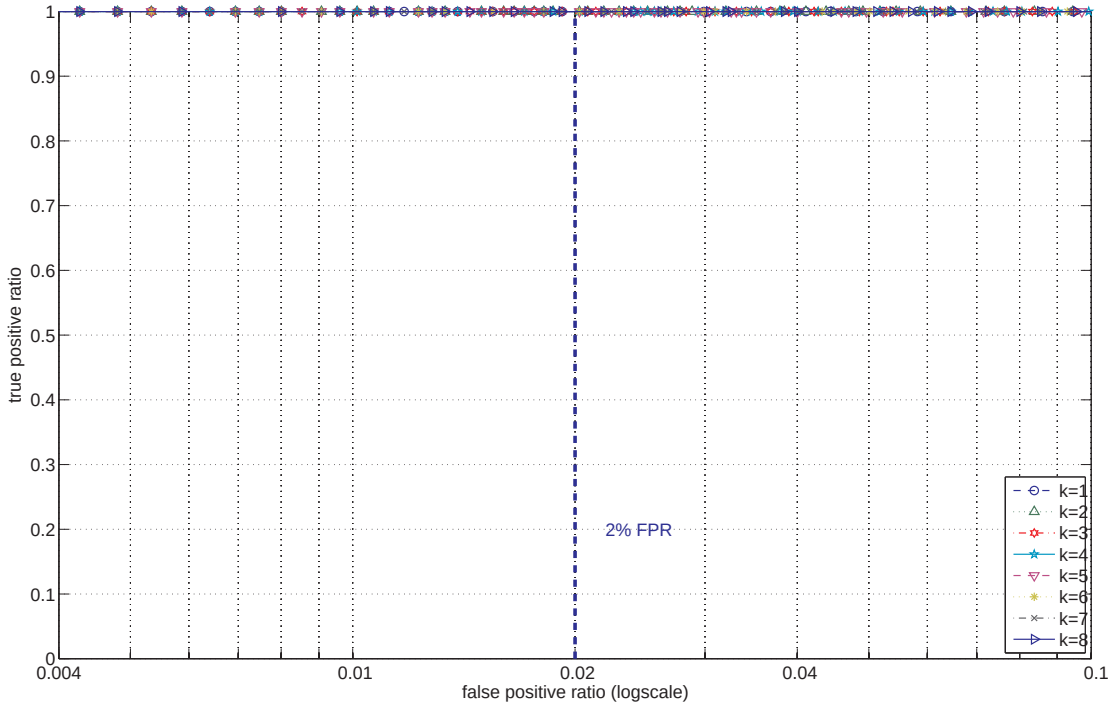


Figure B.18: *mixed*: ROC curves for horizontal port scan with intensity 0.004 without migration

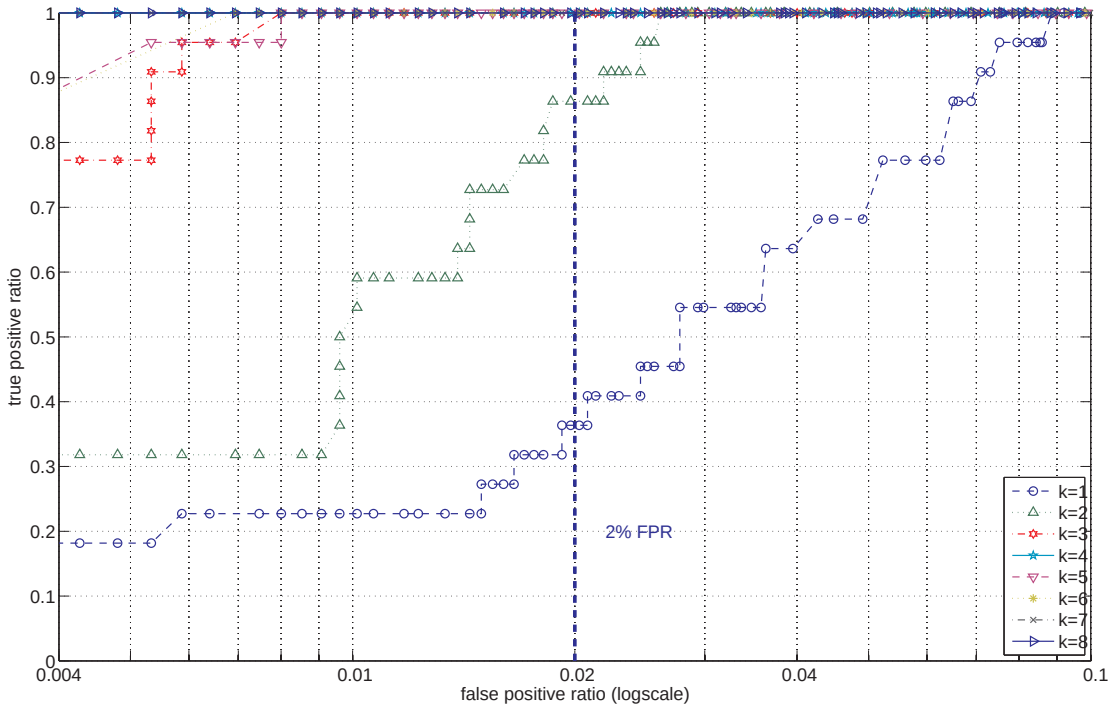


Figure B.19: *alltcp*: ROC curves for vertical port scan with intensity 0.01 without migration

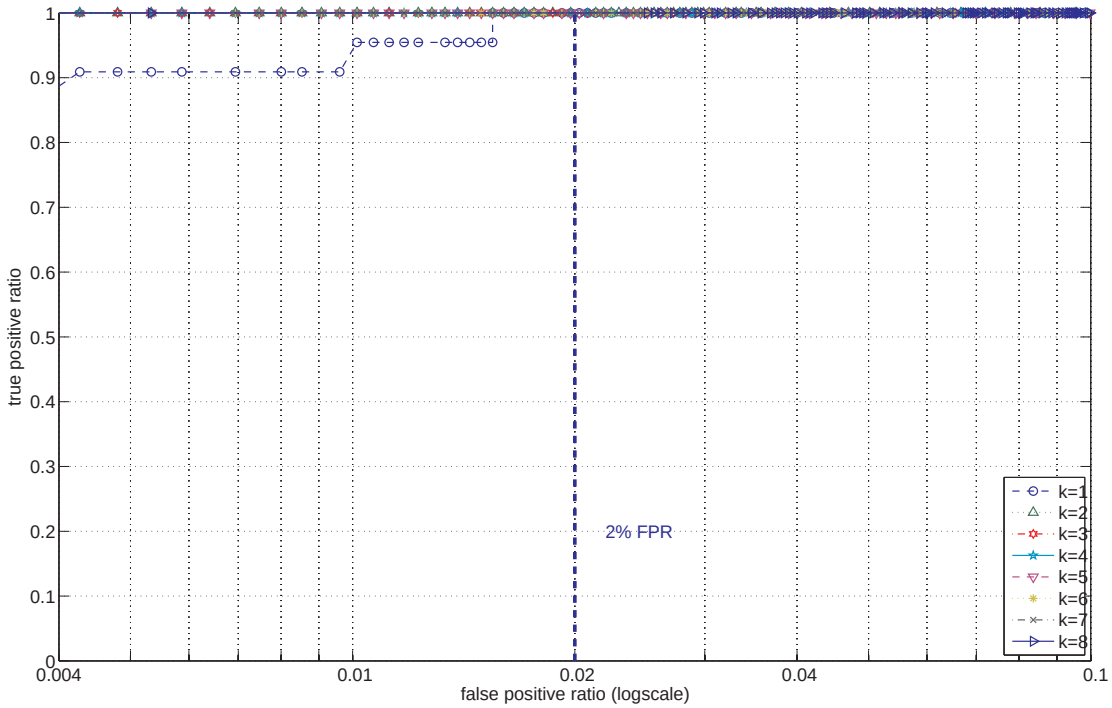


Figure B.20: *alltcp*: ROC curves for vertical port scan with intensity 0.03 without migration

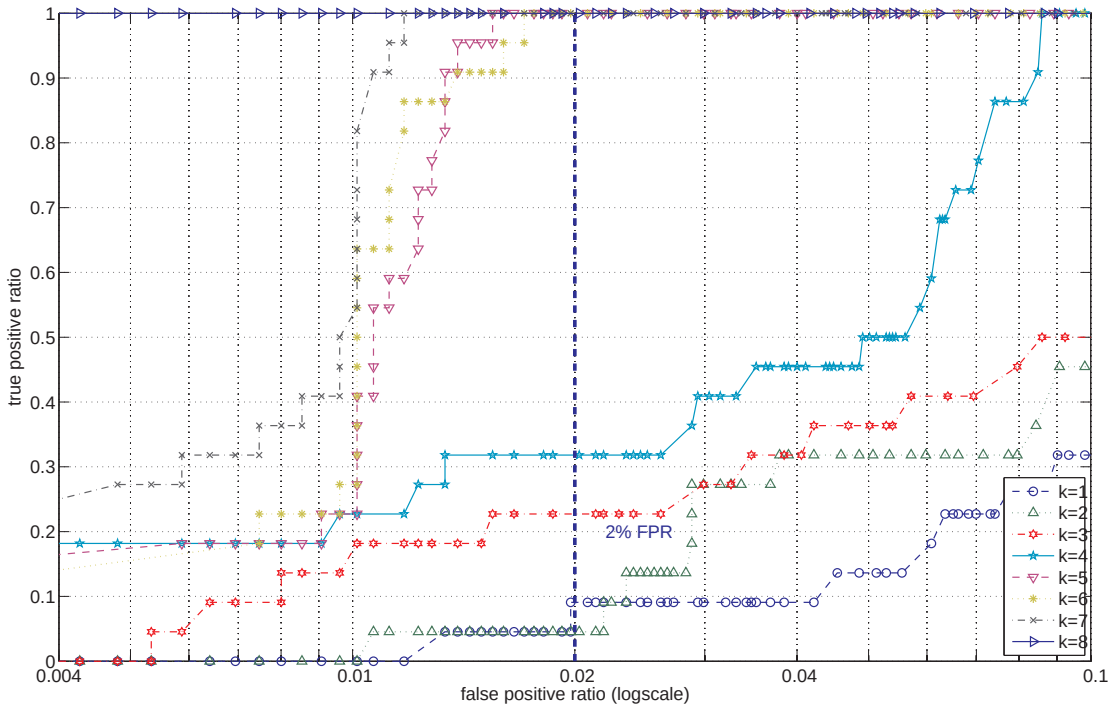


Figure B.21: *alltcp*: ROC curves for horizontal port scan with intensity 0.01 without migration

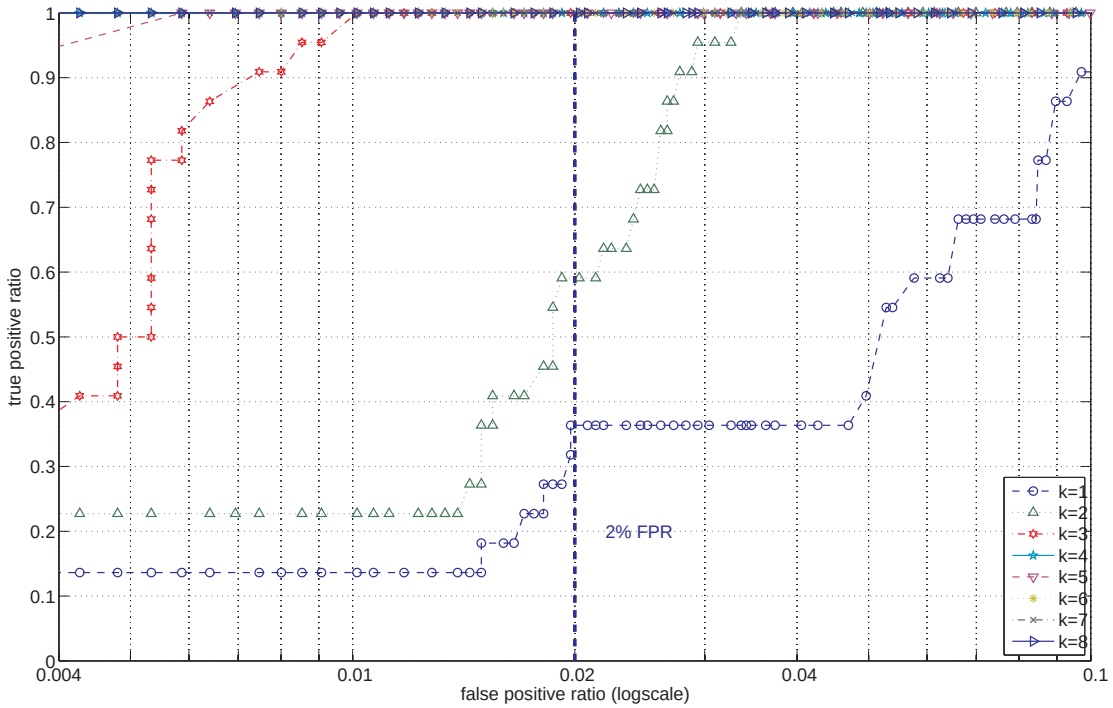


Figure B.22: *alltcp*: ROC curves for horizontal port scan with intensity 0.01 without migration

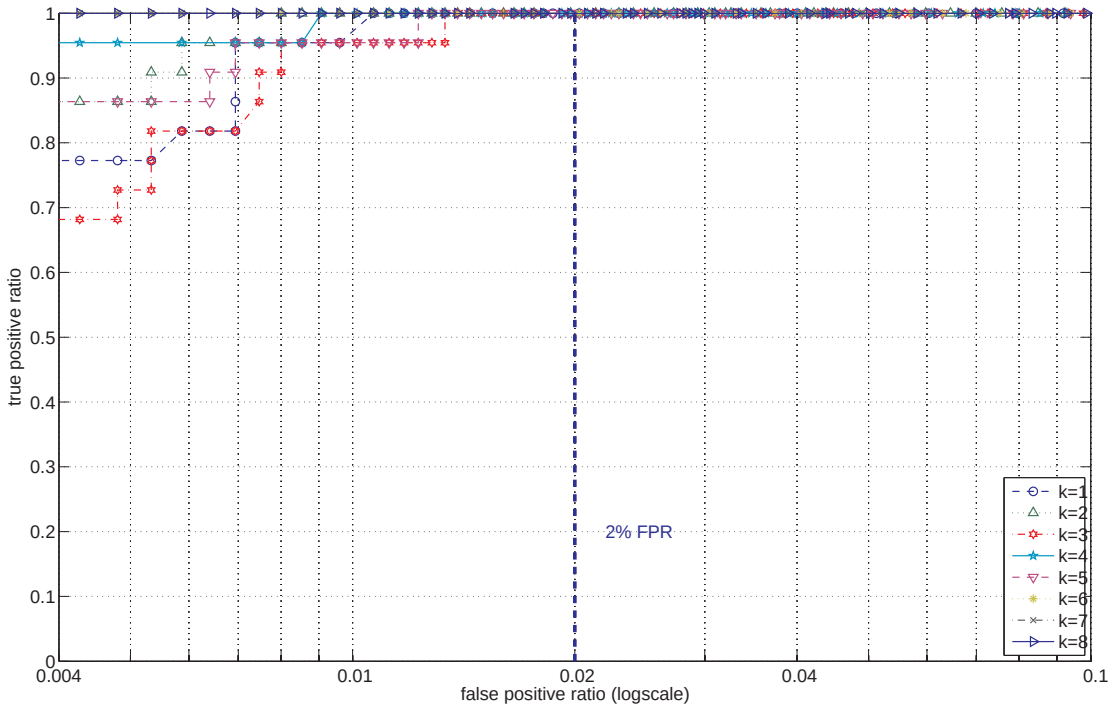


Figure B.23: *mixed*: ROC curves for vertical port scan with intensity 0.001 with 5.8% migration

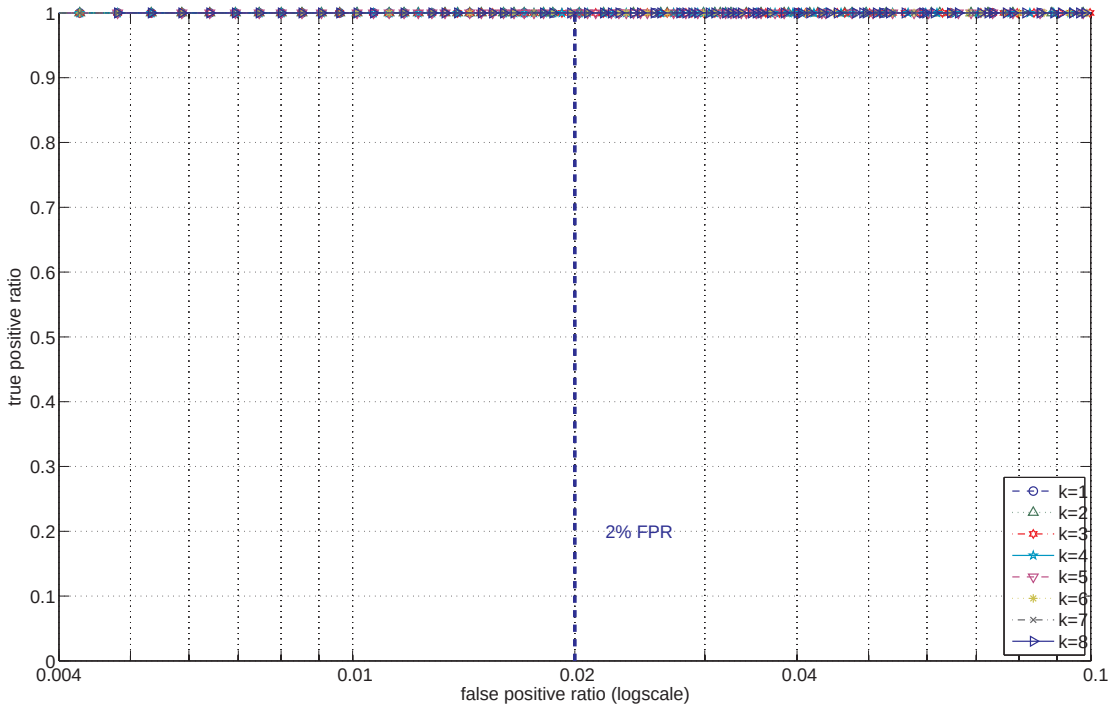


Figure B.24: *mixed*: ROC curves for vertical port scan with intensity 0.004 with 5.8% migration

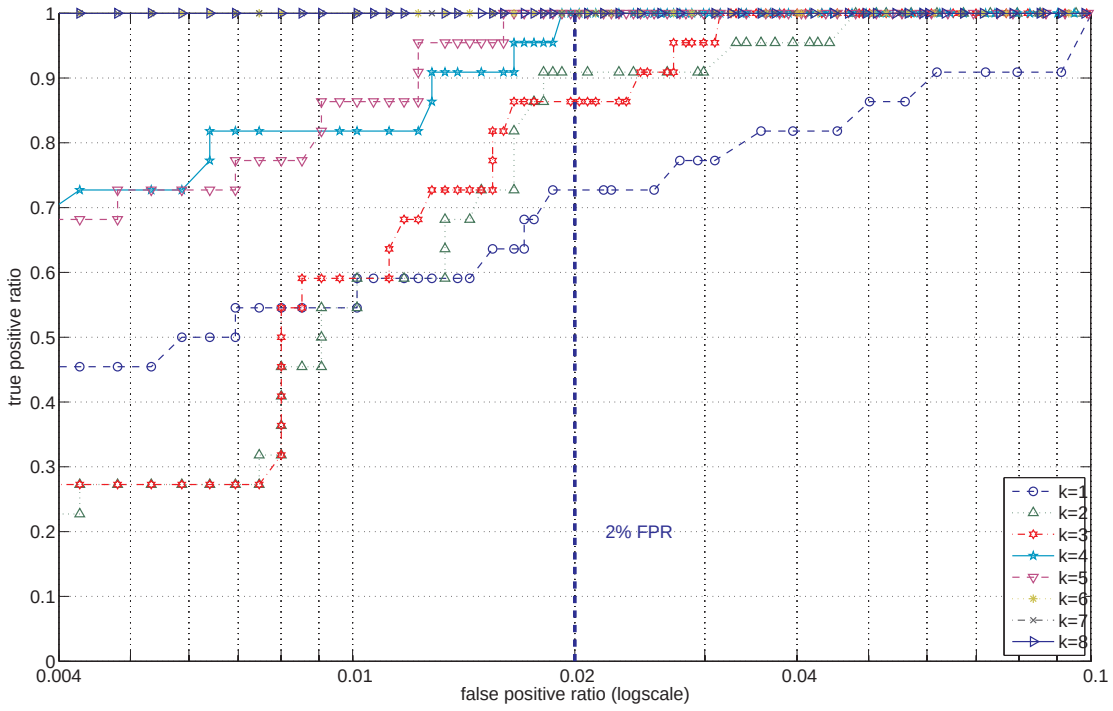


Figure B.25: *mixed*: ROC curves for horizontal port scan with intensity 0.001 with 5.8% migration

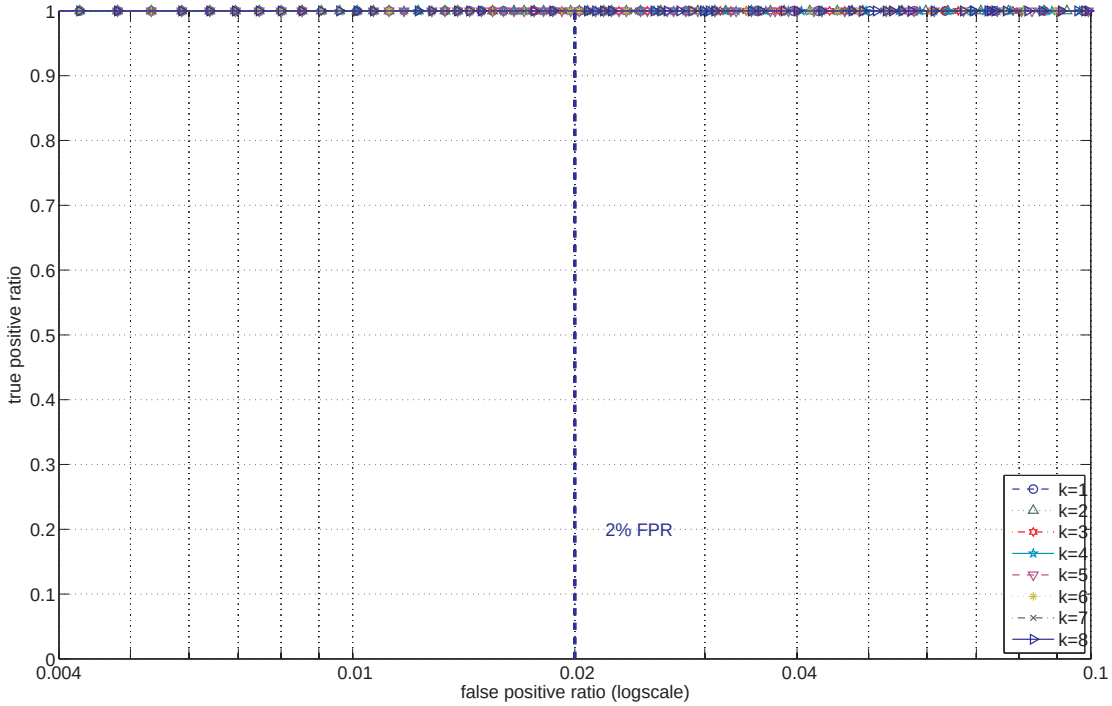


Figure B.26: *mixed*: ROC curves for horizontal port scan with intensity 0.004 with 5.8% migration

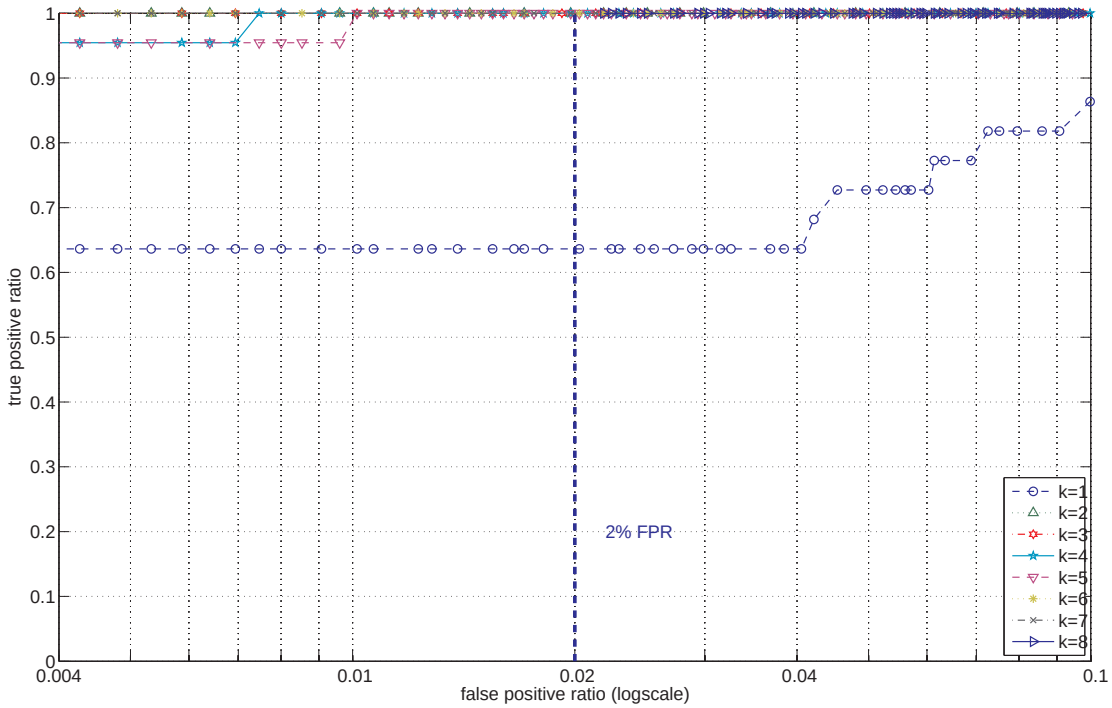


Figure B.27: *mixed*: ROC curves for vertical port scan with intensity 0.001 with 30% migration

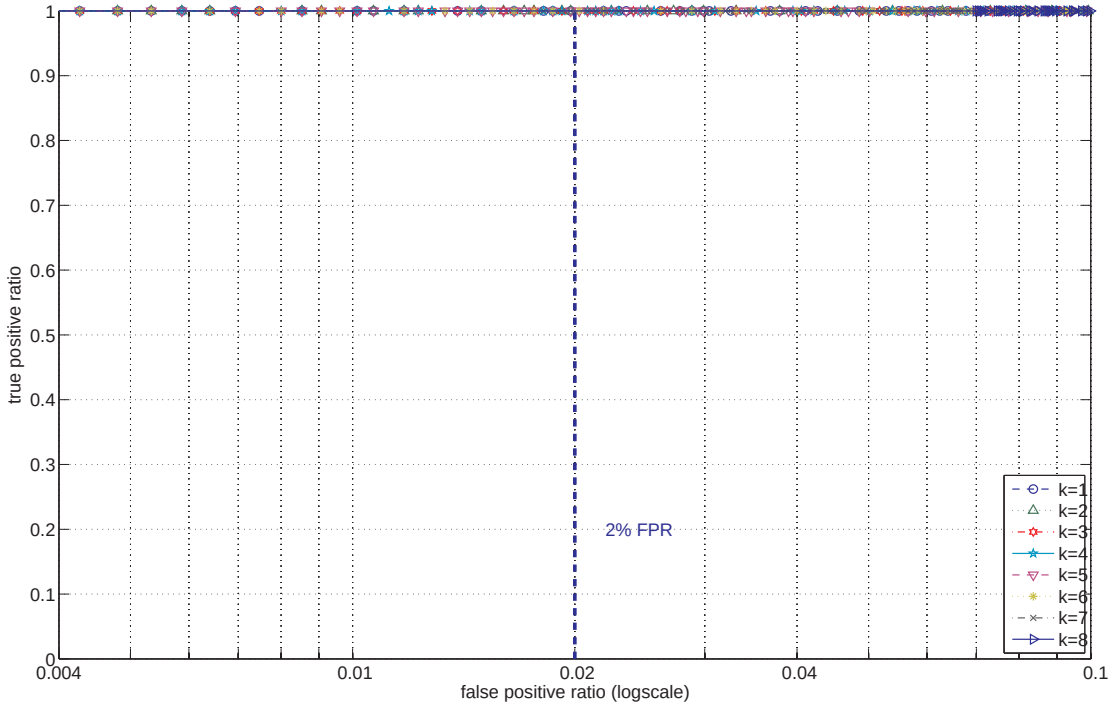


Figure B.28: *mixed*: ROC curves for vertical port scan with intensity 0.004 with 30% migration

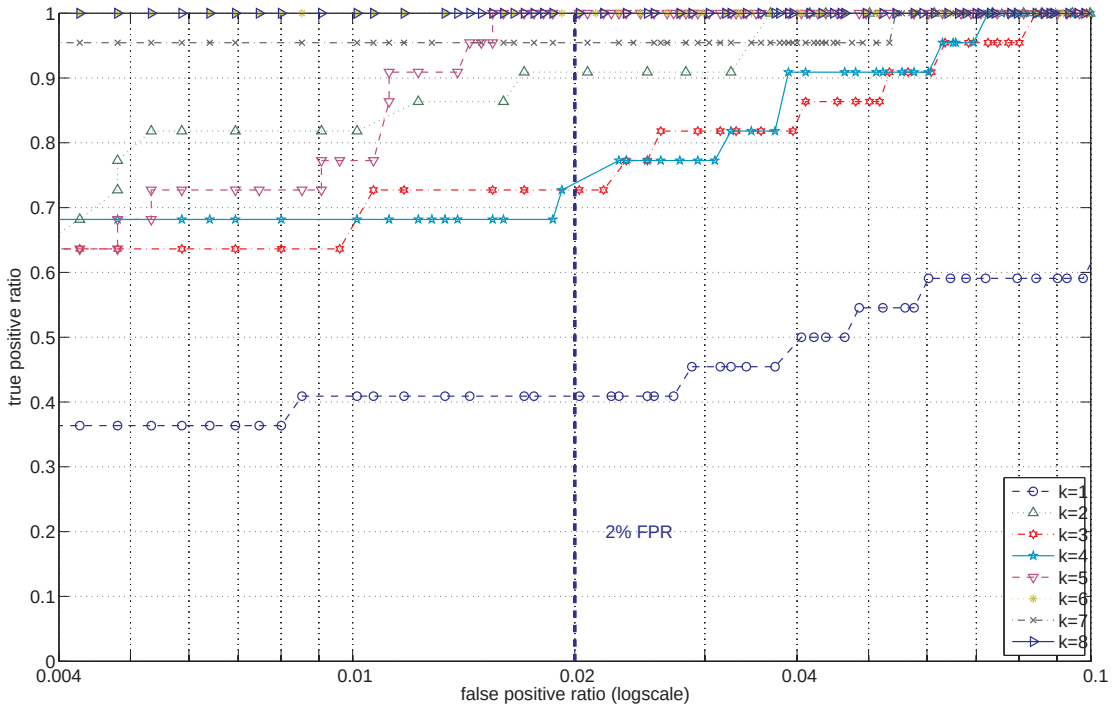


Figure B.29: *mixed*: ROC curves for horizontal port scan with intensity 0.001 with 30% migration

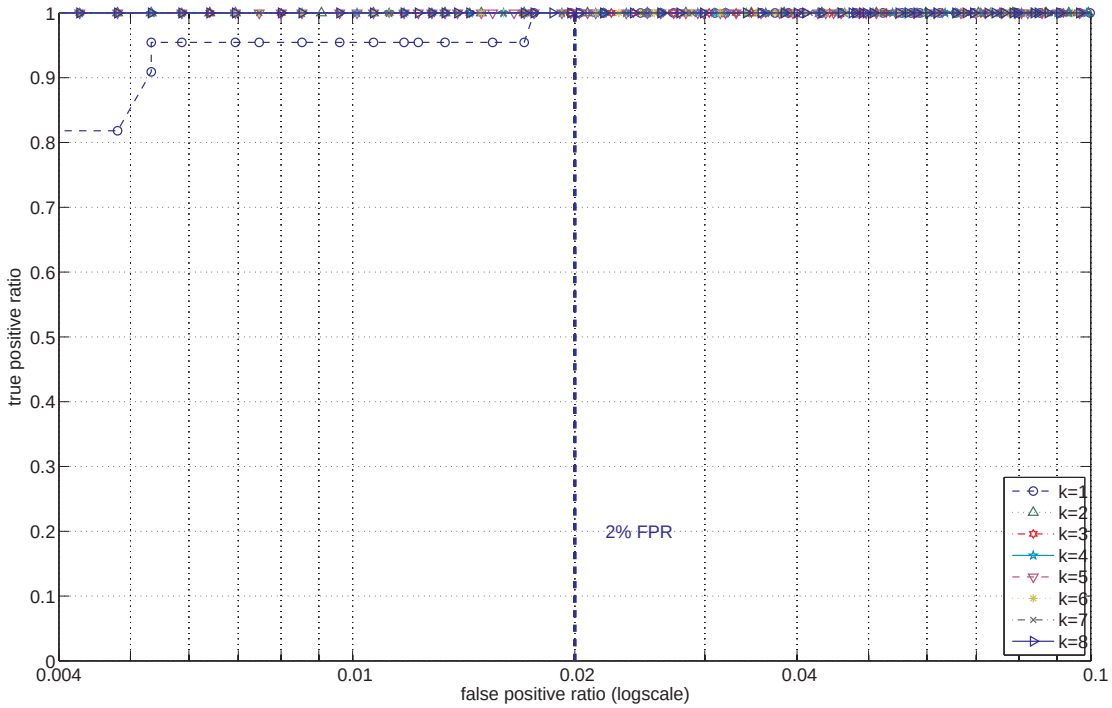


Figure B.30: *mixed*: ROC curves for horizontal port scan with intensity 0.004 with 30% migration

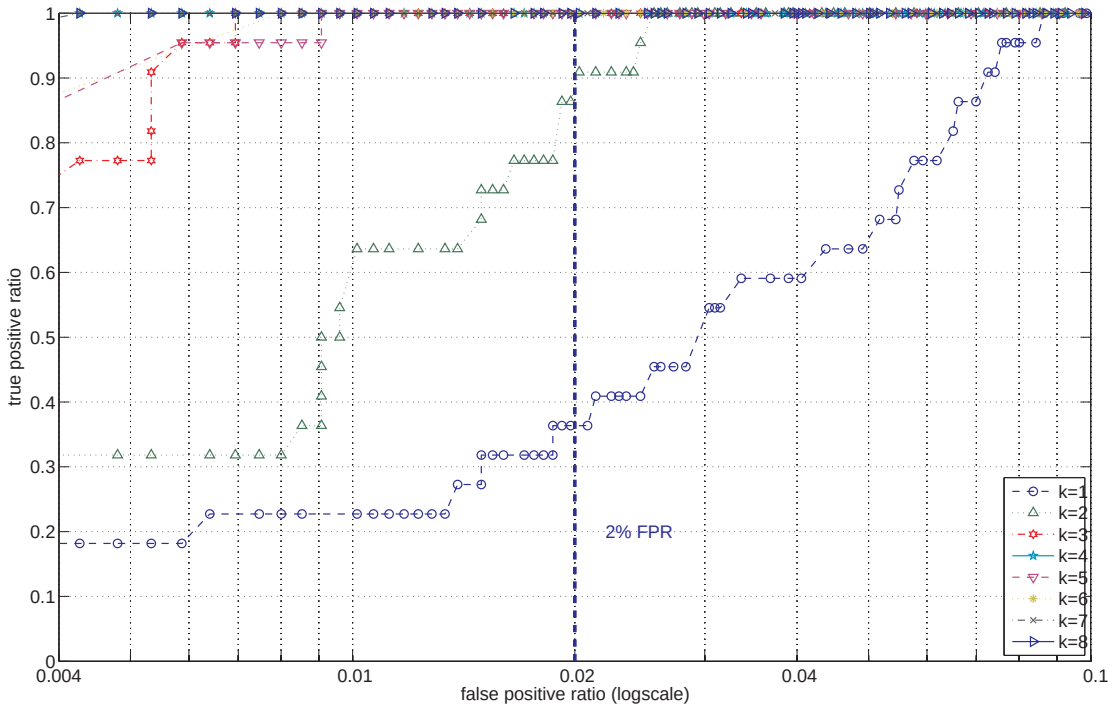


Figure B.31: *alltcp*: ROC curves for vertical port scan with intensity 0.01 with 2.5% migration

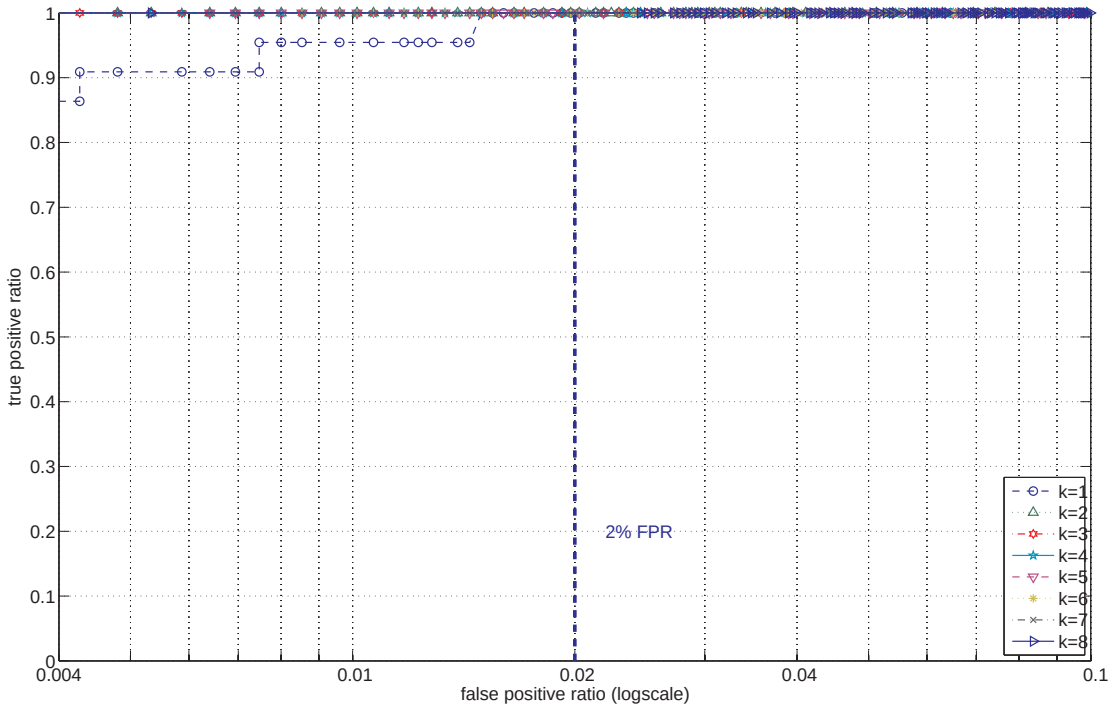


Figure B.32: *alltcp*: ROC curves for vertical port scan with intensity 0.03 with 2.5% migration

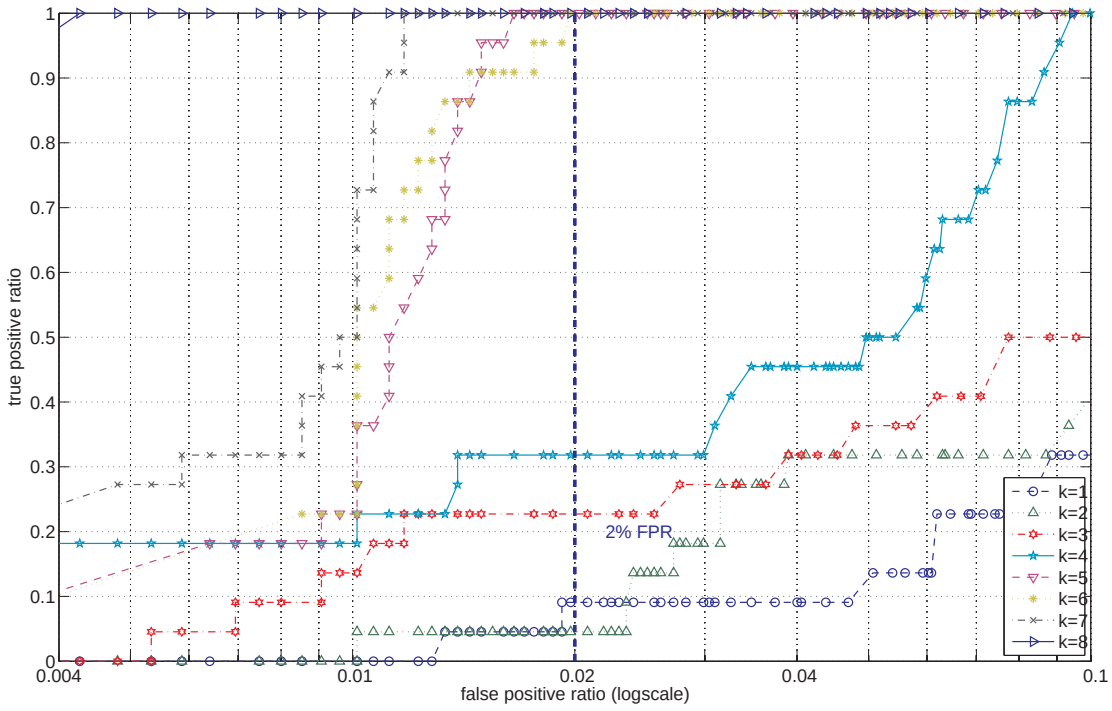


Figure B.33: *alltcp*: ROC curves for horizontal port scan with intensity 0.01 with 2.5% migration

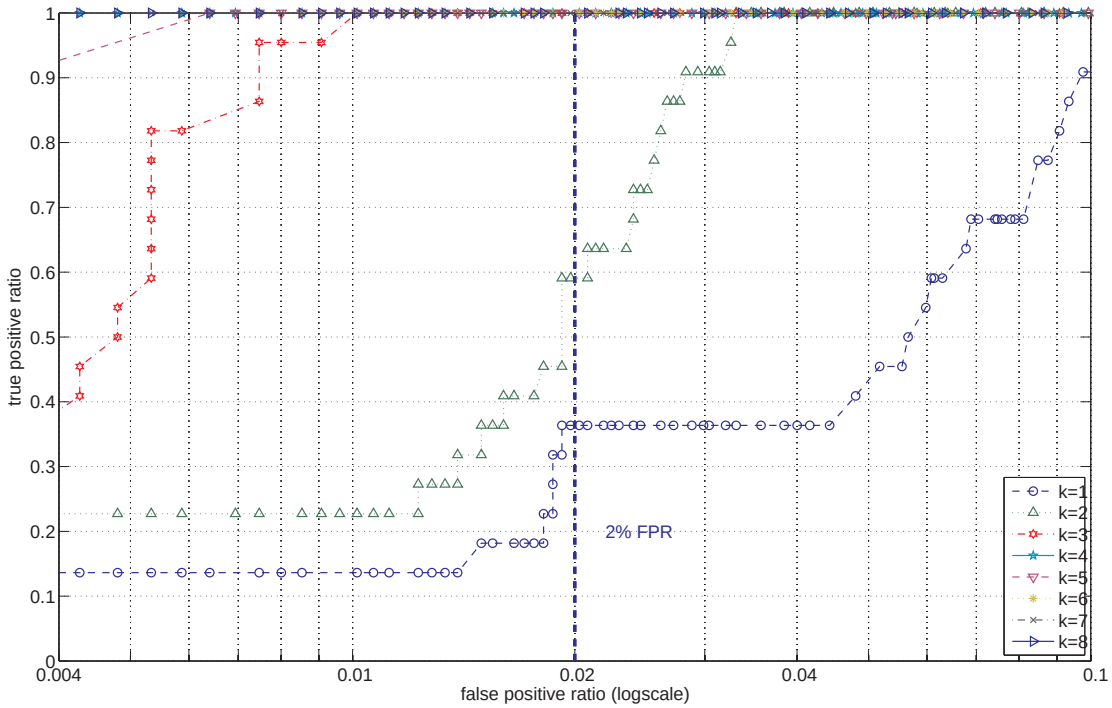


Figure B.34: *alltcp*: ROC curves for horizontal port scan with intensity 0.01 with 2.5% migration

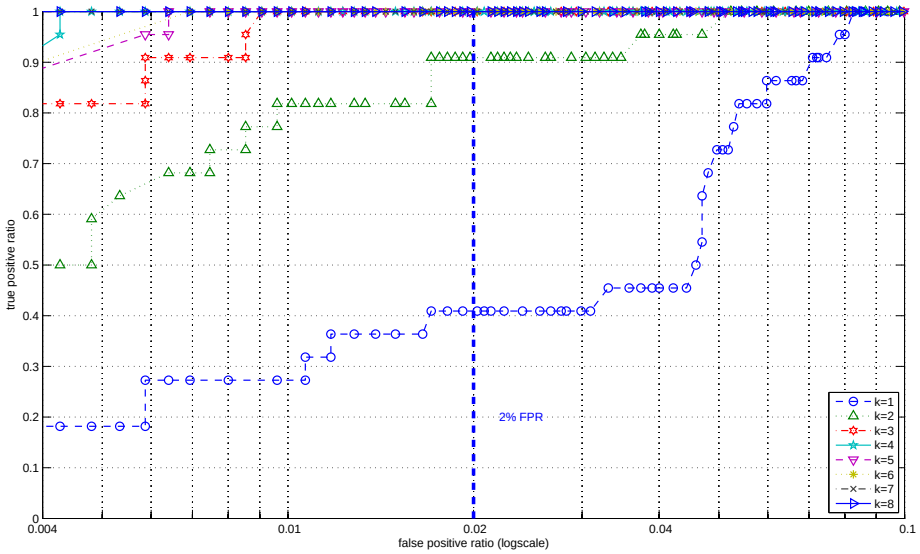


Figure B.35: *alltcp*: ROC curves for vertical port scan with intensity 0.01 with 10% migration

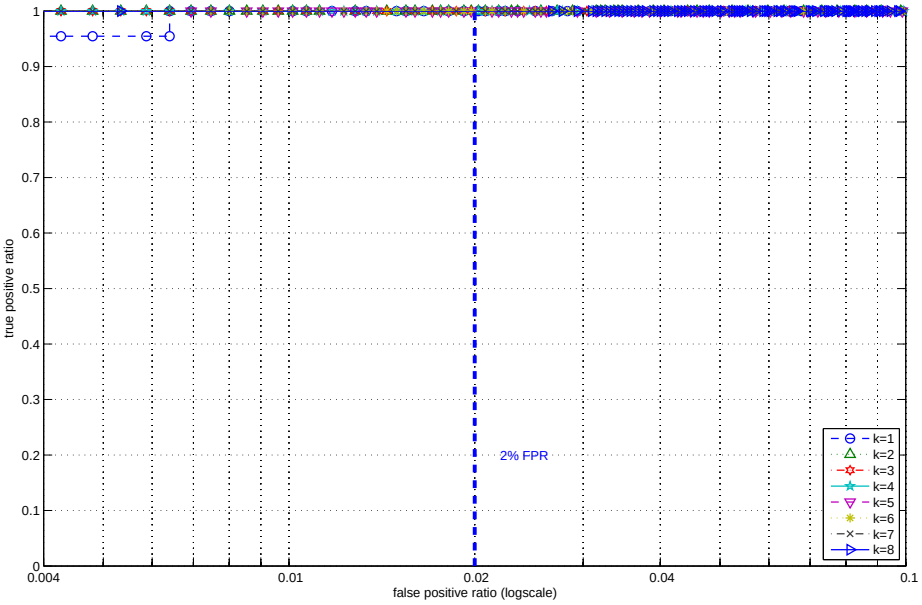


Figure B.36: *alltcp*: ROC curves for vertical port scan with intensity 0.03 with 10% migration

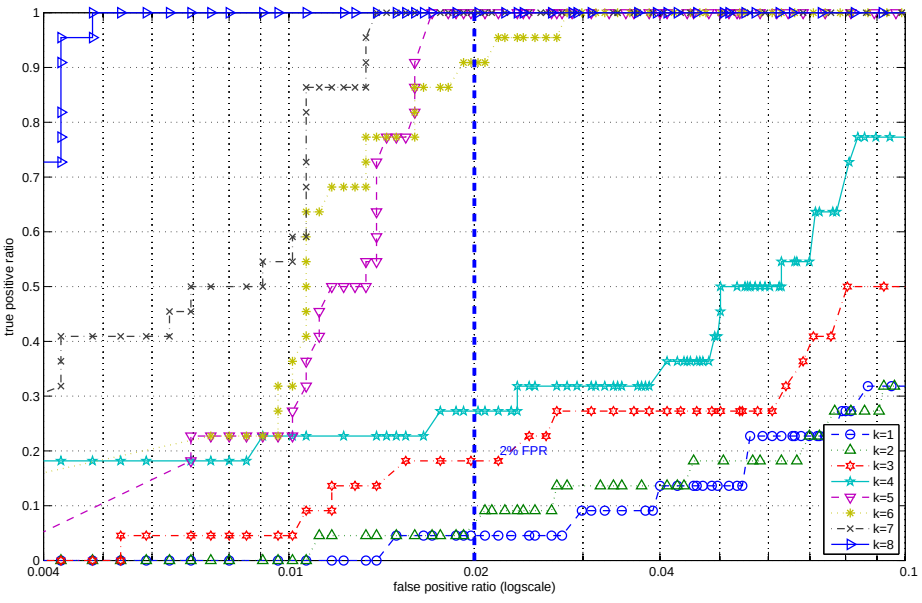


Figure B.37: *alltcp*: ROC curves for horizontal port scan with intensity 0.01 with 10% migration

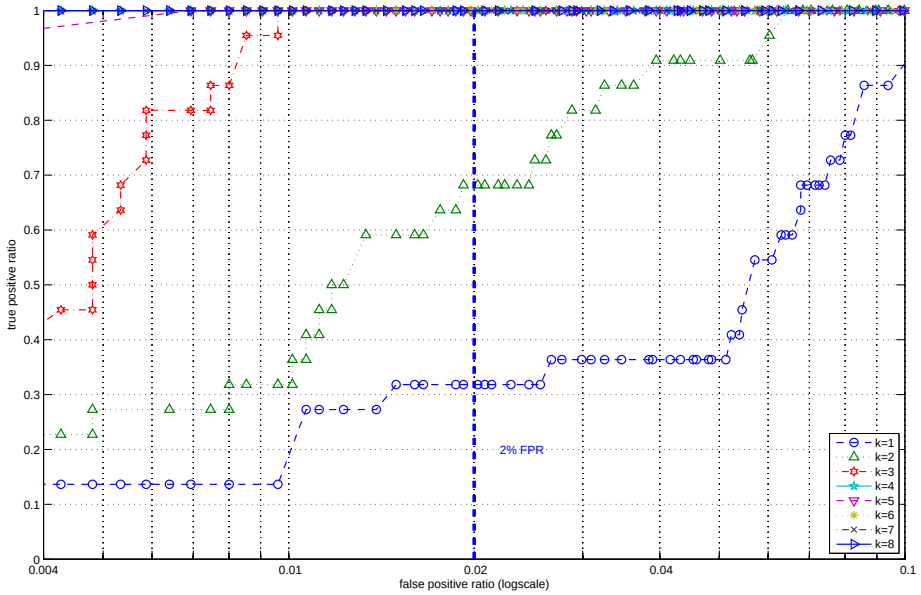


Figure B.38: *alltcp*: ROC curves for horizontal port scan with intensity 0.03 with 10% migration

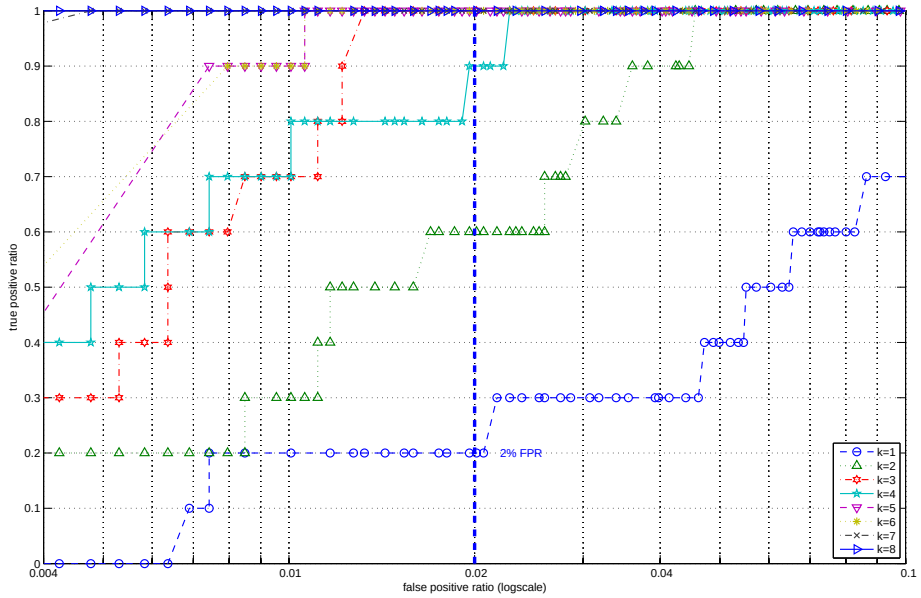


Figure B.39: *alltcp*: ROC curves for DDoS with intensity 0.03 with 10% migration

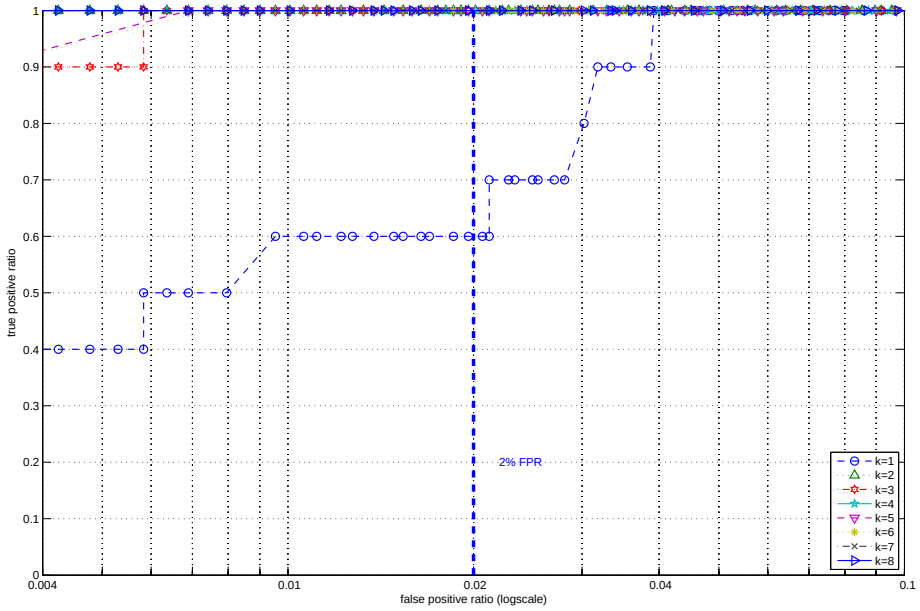


Figure B.40: *alltcp*: ROC curves for DDoS with intensity 0.05 with 10% migration

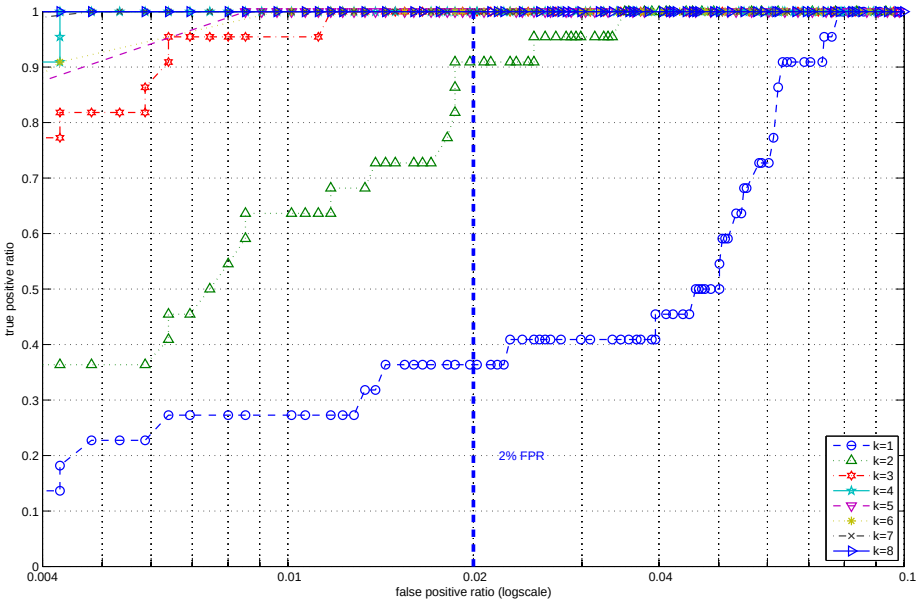


Figure B.41: *alltcp*: ROC curves for vertical port scan with intensity 0.01 with 21% migration

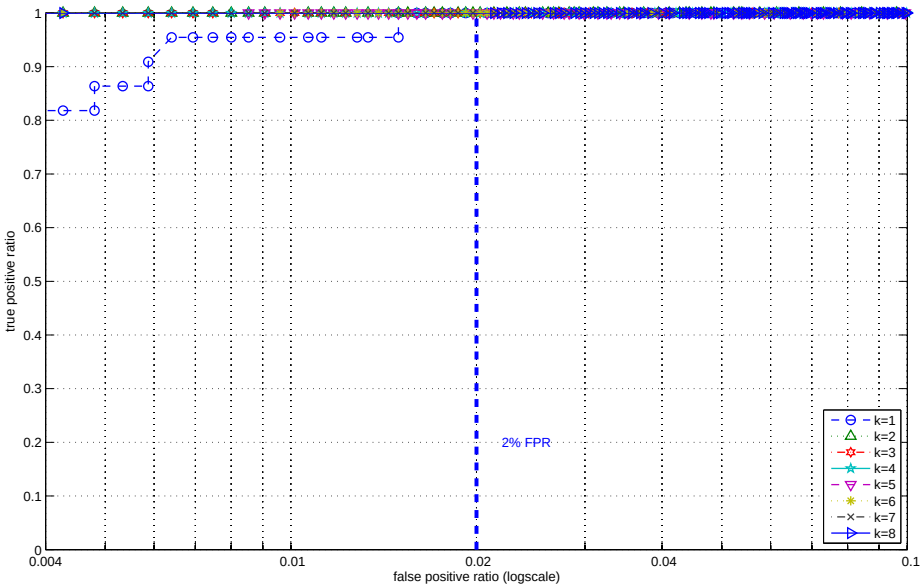


Figure B.42: *alltcp*: ROC curves for vertical port scan with intensity 0.03 with 21% migration

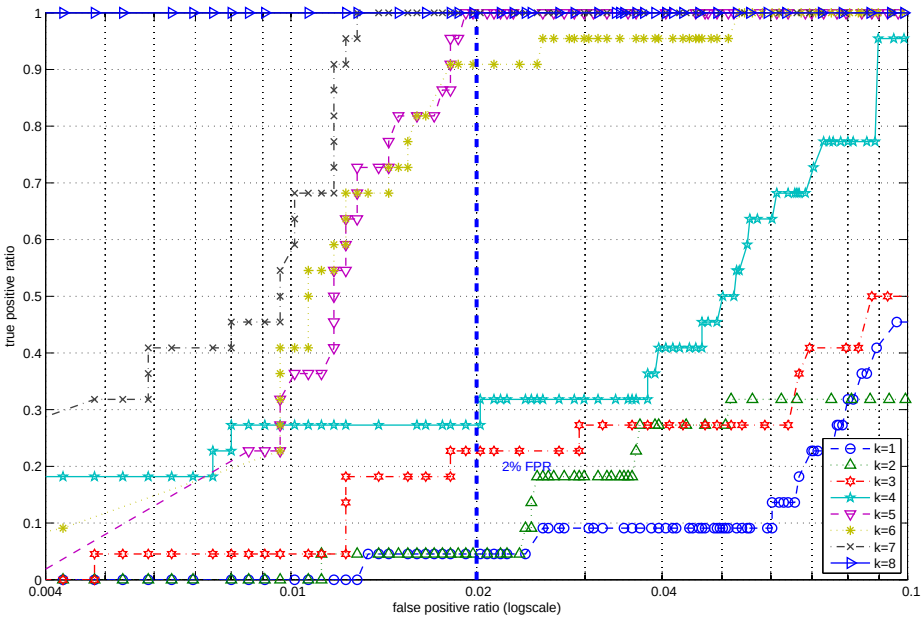


Figure B.43: *alltcp*: ROC curves for horizontal port scan with intensity 0.01 with 21% migration

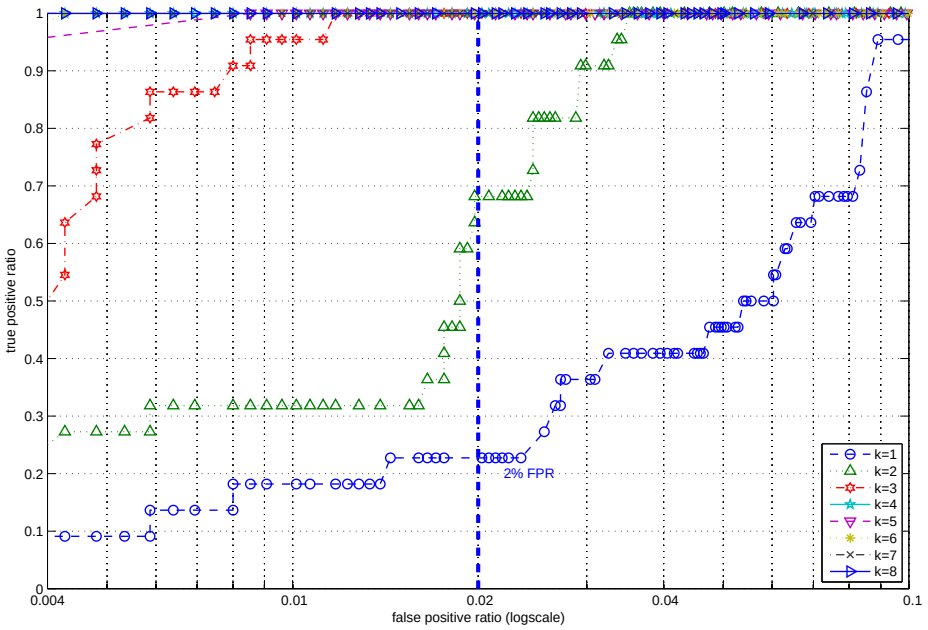


Figure B.44: *alltcp*: ROC curves for horizontal port scan with intensity 0.03 with 21% migration

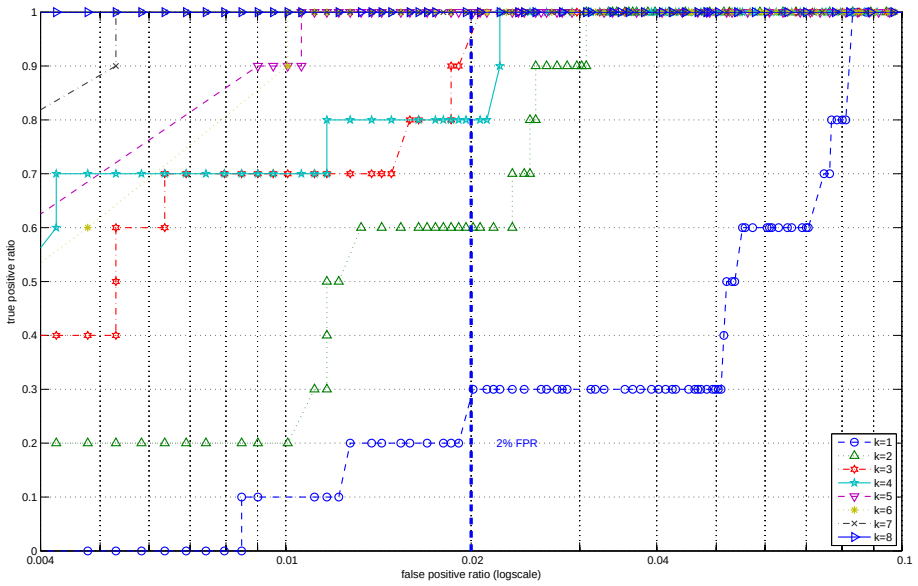


Figure B.45: *alltcp*: ROC curves for DDoS with intensity 0.03 with 21% migration

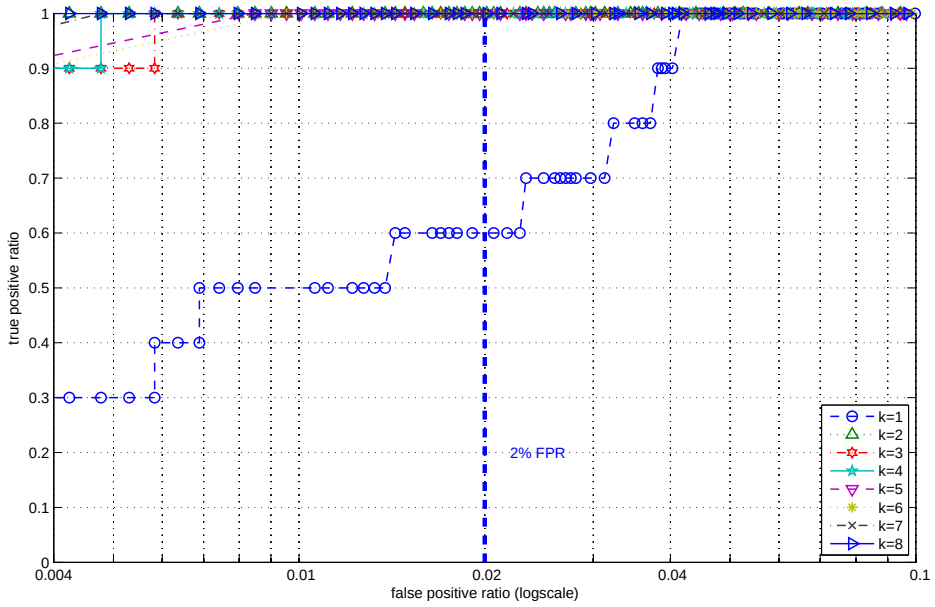


Figure B.46: *alltcp*: ROC curves for DDoS with intensity 0.05 with 21% migration

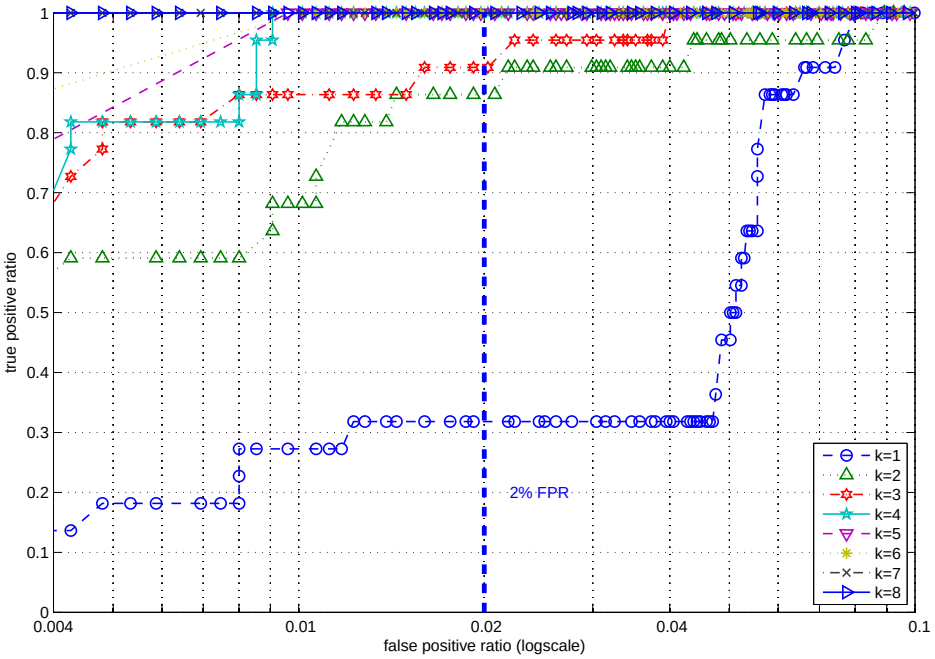


Figure B.47: *alltcp*: ROC curves for vertical port scan with intensity 0.01 with 28% migration

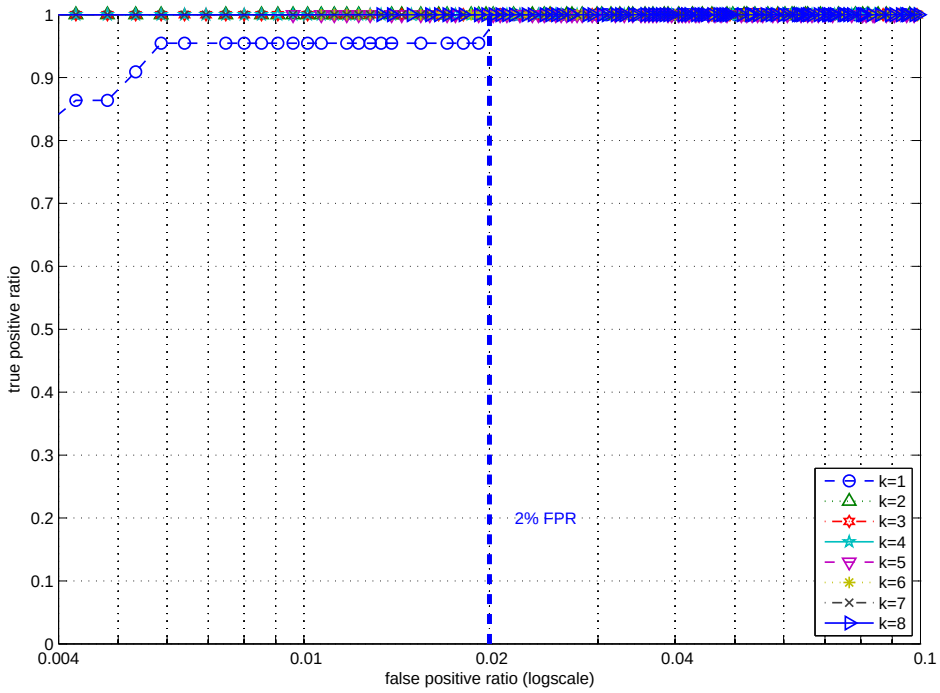


Figure B.48: *alltcp*: ROC curves for vertical port scan with intensity 0.03 with 28% migration

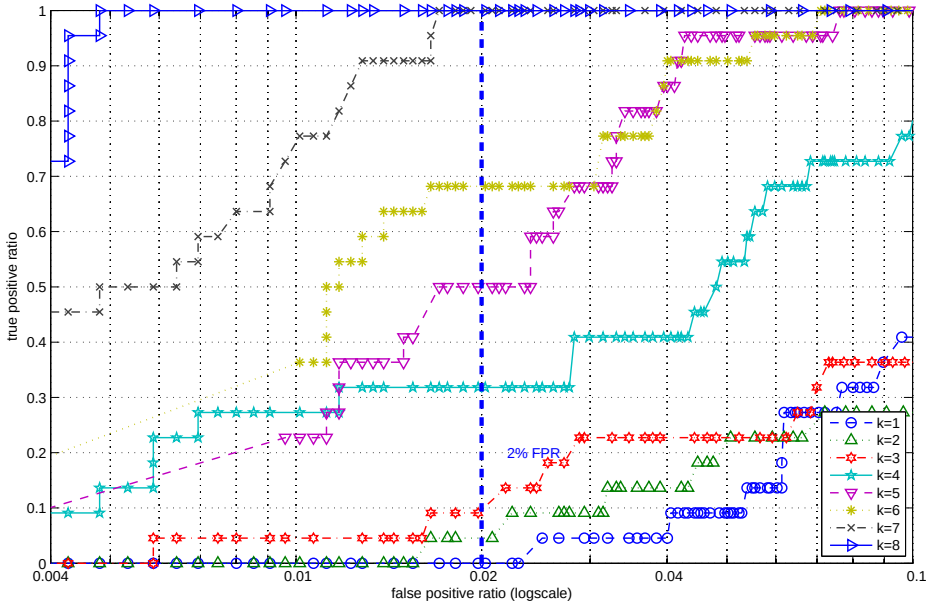


Figure B.49: *alltcp*: ROC curves for horizontal port scan with intensity 0.01 with 28% migration

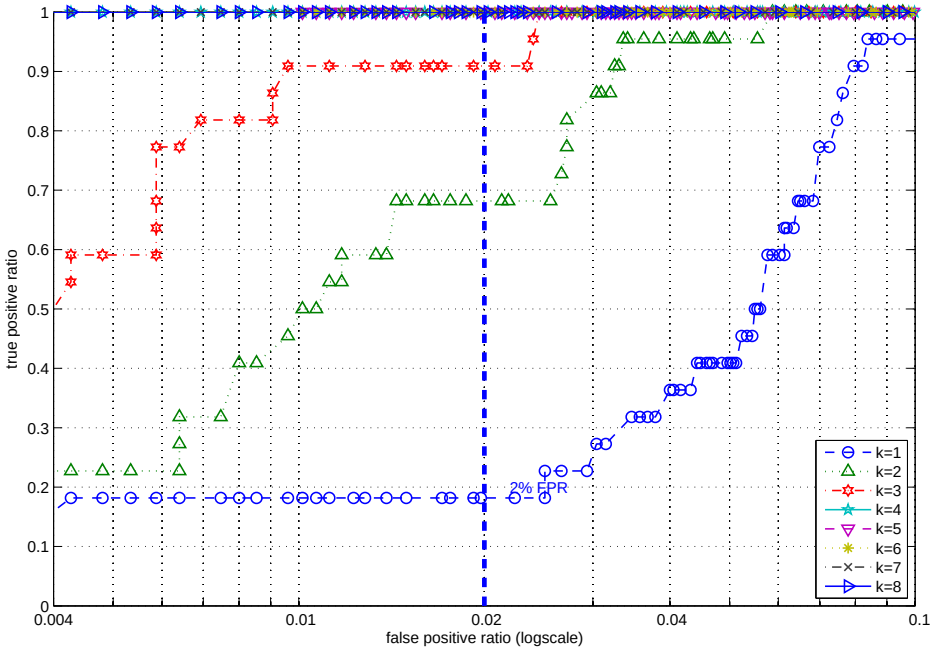


Figure B.50: *alltcp*: ROC curves for horizontal port scan with intensity 0.03 with 28% migration

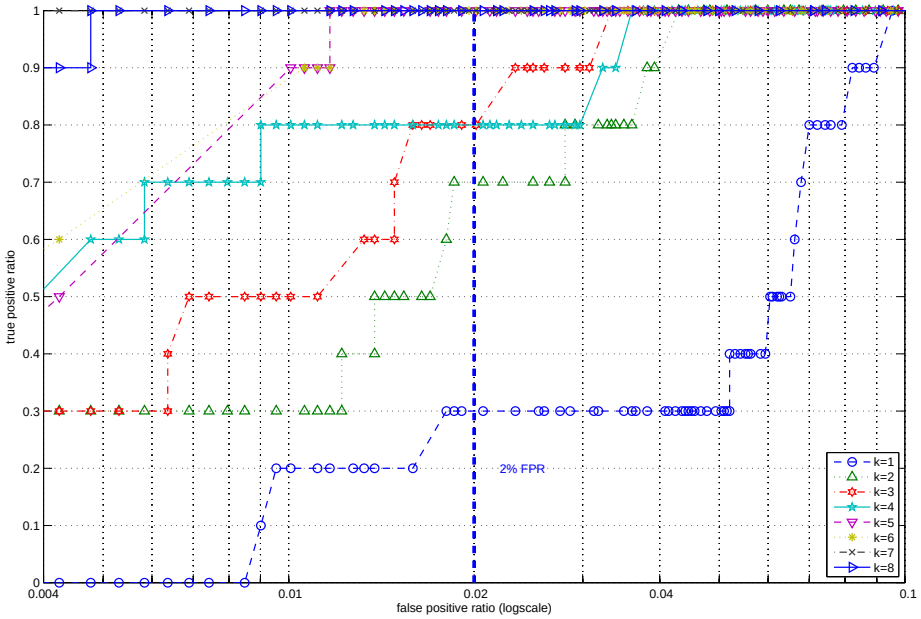


Figure B.51: *alltcp*: ROC curves for DDoS with intensity 0.03 with 28% migration

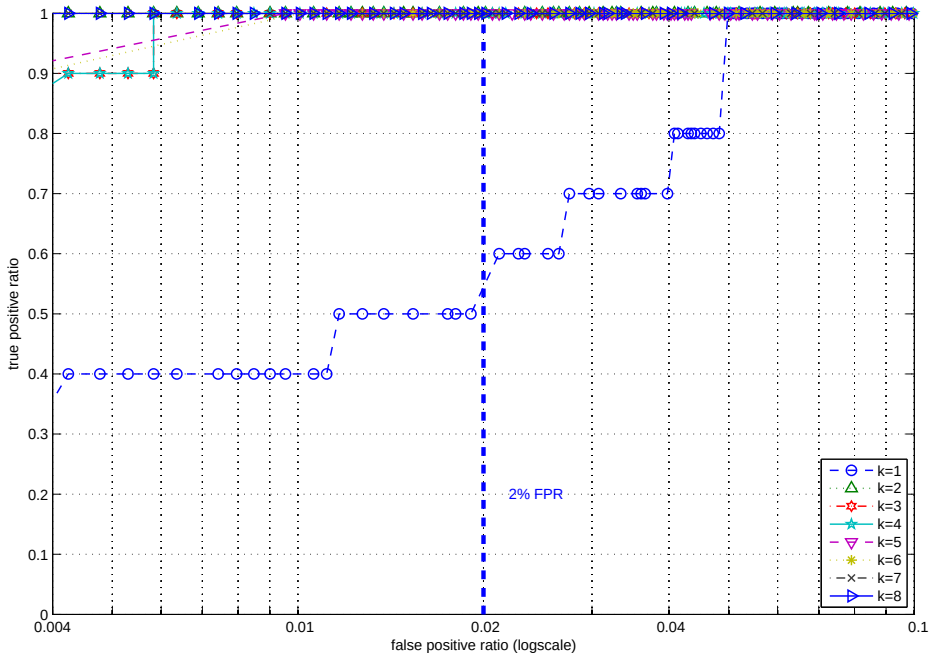


Figure B.52: *alltcp*: ROC curves for DDoS with intensity 0.05 with 28% migration

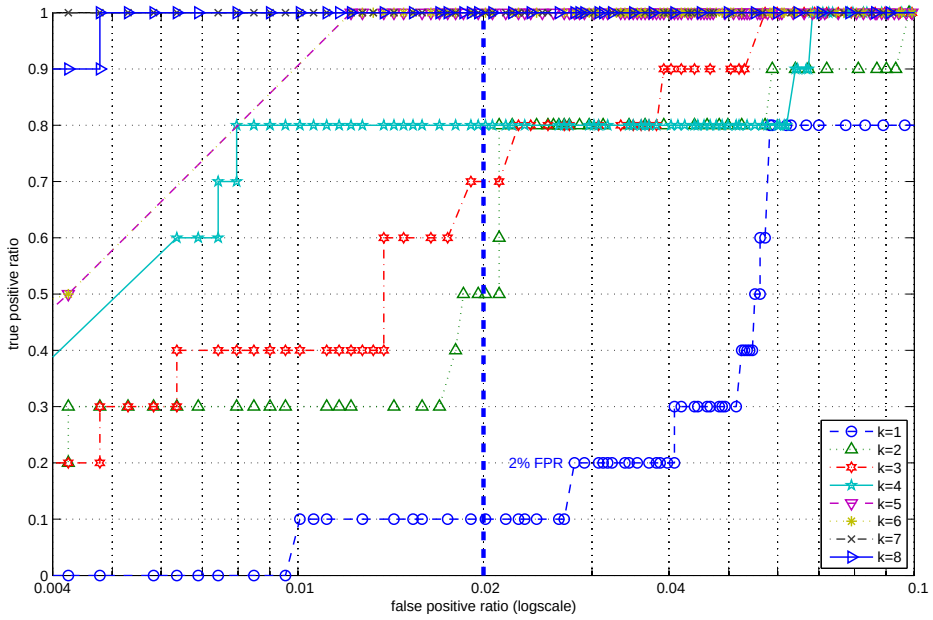


Figure B.53: *alltcp*: ROC curves for DDoS with intensity 0.03 with 39% migration

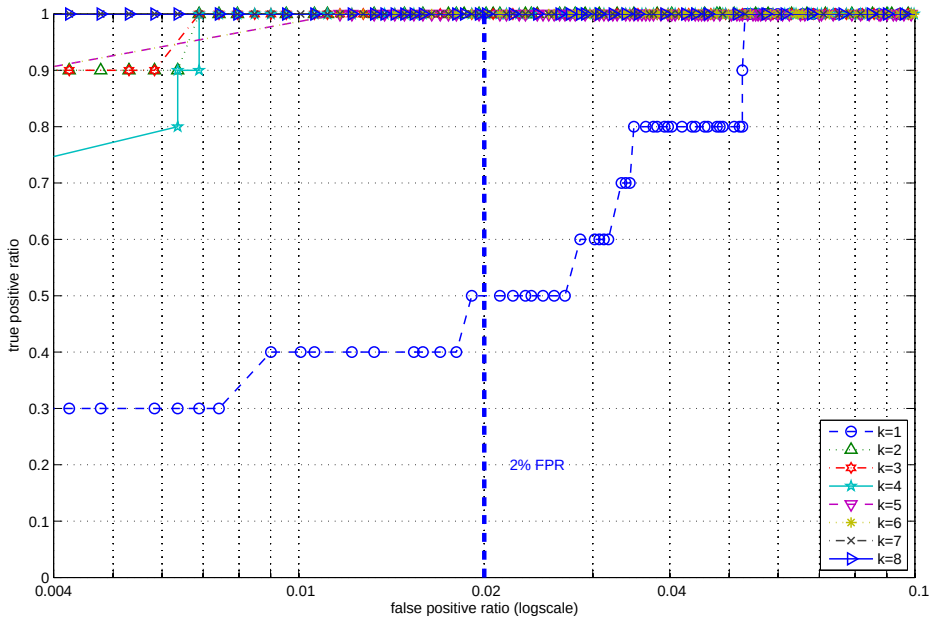


Figure B.54: *alltcp*: ROC curves for DDoS with intensity 0.05 with 39% migration

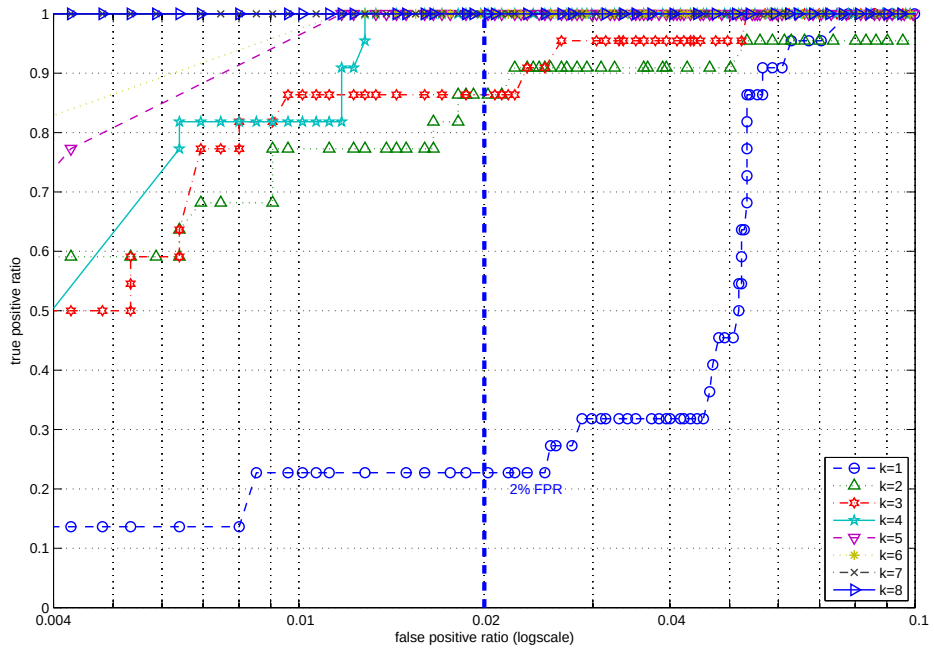


Figure B.55: *alltcp*: ROC curves for vertical port scan with intensity 0.01 with 39% migration

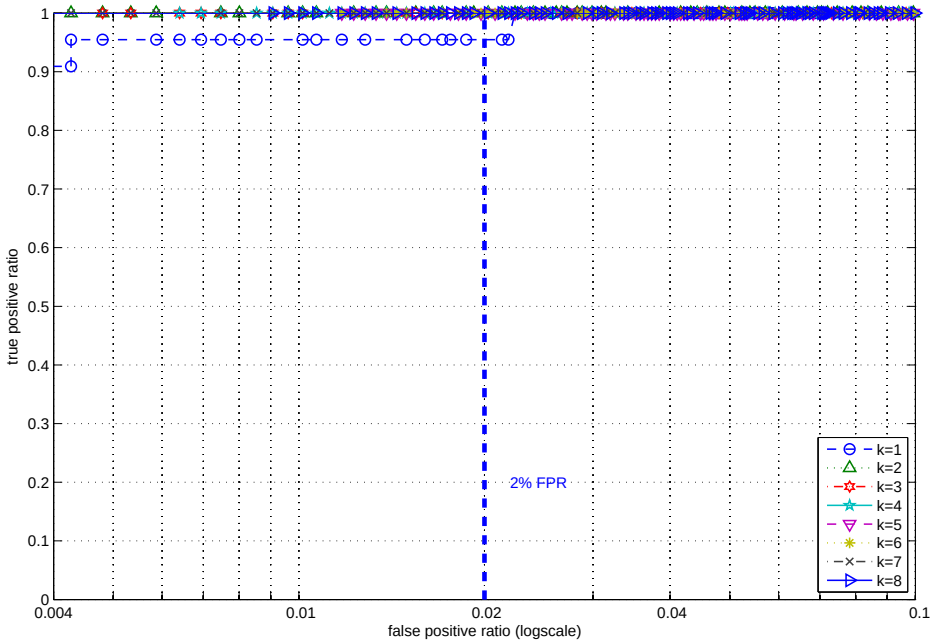


Figure B.56: *alltcp*: ROC curves for vertical port scan with intensity 0.03 with 39% migration

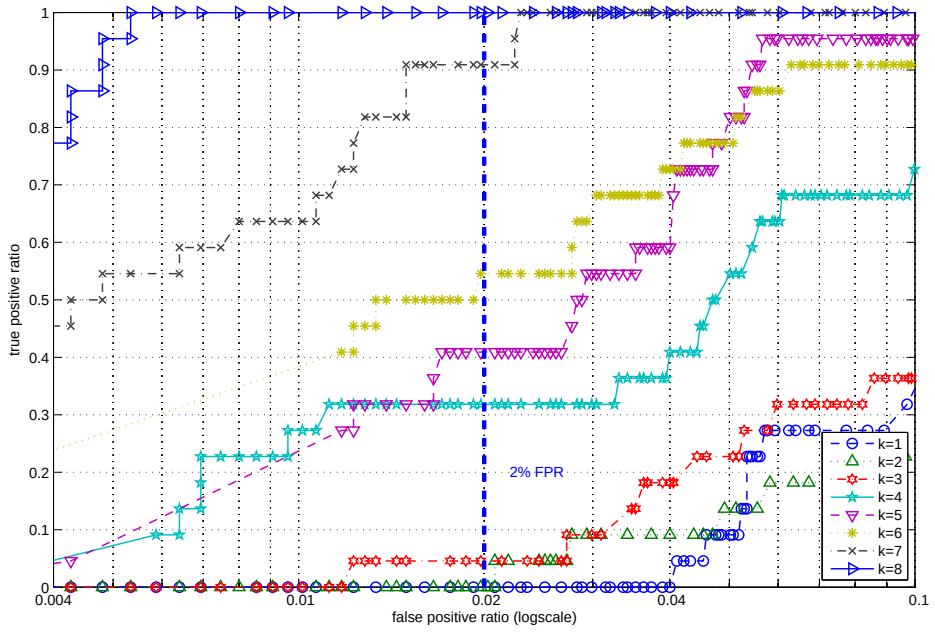


Figure B.57: *alltcp*: ROC curves for horizontal port scan with intensity 0.01 with 39% migration

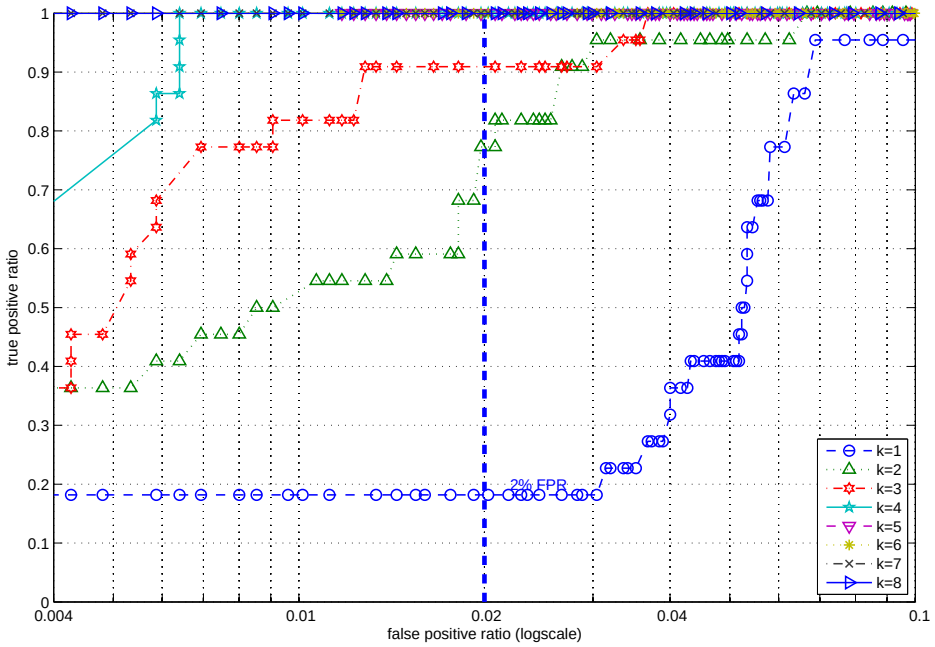


Figure B.58: *alltcp*: ROC curves for horizontal port scan with intensity 0.03 with 39% migration