Intelligent Information Management, 2015, 7, 283-302
Published Online November 2015 in SciRes. http://www.scirp.org/journal/iim
http://dx.doi.org/10.4236/iim.2015.76023

Real-Time Timing Channel Detection in a Software-Defined Networking Virtual Environment

Anyi Liu[1], Jim X. Chen[2], Harry Wechsler[2]

[1]Department of Computer Science, Indiana University—Purdue University Fort Wayne, Fort Wayne, USA
[2]Department of Computer Science, George Mason University, Fairfax, USA
Email: liua@ipfw.edu, jchen@gmu.edu, wechsler@cs.gmu.edu

Received 3 September 2015; accepted 9 November 2015; published 12 November 2015

Abstract

Despite extensive research, timing channels (TCs) are still known as a principal category of threats that aim to leak and transmit information by perturbing the timing or ordering of events. Existing TC detection approaches use either signature-based approaches to detect known TCs or anomaly-based approach by modeling the legitimate network traffic in order to detect unknown TCs. Unfortunately, in a software-defined networking (SDN) environment, most existing TC detection approaches would fail due to factors such as volatile network traffic, imprecise timekeeping mechanisms, and dynamic network topology. Furthermore, stealthy TCs can be designed to mimic the legitimate traffic pattern and thus evade anomalous TC detection. In this paper, we overcome the above challenges by presenting a novel framework that harnesses the advantages of elastic resources in the cloud. In particular, our framework dynamically configures SDN to enable/disable differential analysis against outbound network flows of different virtual machines (VMs). Our framework is tightly coupled with a new metric that first decomposes the timing data of network flows into a number of using the *discrete wavelet-based multi-resolution transform* (DWMT). It then applies the *Kullback-Leibler divergence* (KLD) to measure the variance among flow pairs. The appealing feature of our approach is that, compared with the existing anomaly detection approaches, it can detect most existing and some new stealthy TCs without legitimate traffic for modeling, even with the presence of noise and imprecise timekeeping mechanism in an SDN virtual environment. We implement our framework as a prototype system, OBSERVER, which can be dynamically deployed in an SDN environment. Empirical evaluation shows that our approach can efficiently detect TCs with a higher detection rate, lower latency, and negligible performance overhead compared to existing approaches.

Keywords

Covert Channel, Timing Channel, Intrusion Detection, Virtualization, Software-Defined

How to cite this paper: Liu, A.Y., Chen, J.X. and Wechsler, H. (2015) Real-Time Timing Channel Detection in a Software-Defined Networking Virtual Environment. *Intelligent Information Management*, **7**, 283-302.
http://dx.doi.org/10.4236/iim.2015.76023

Network

1. Introduction

The widespread deployment of firewalls and other perimeter defenses to protect information in enterprise information systems in the past decade has raised the bar for malicious outside adversary breaching a well-protected network. However, the same enterprise can still be easily subverted by malicious insiders, who can potentially exfiltrate inside secrete through a special communication channel, namely *covert channels* (CC) [1]-[4]. A successful CC can exfiltrate inside secrete by modifying a "storage location" (namely *storage channel* or SC) [5], or manipulating timing or ordering of events (namely *timing channel* or TC) [5], without triggering any alert in a well-protected network [6]. With the increasing number of TC design schemes [7] [8], defending against TCs is important and quite challenging.

Most TC detection approaches detect TCs by using either signature-based approaches to detect known TCs [9] or anomaly-based approaches by modeling legitimate network traffic in order to detect unknown TCs [5] [6] [10] [11]. The existing detection approaches have proven to be efficient in a network environment, where actual computers are physically connected and the network topology is relatively stable. However, most of the TC detection schemes face at least two challenges when the problem space is moved to an SDN virtual environment [12]. First, compared to the traditional network environment, where legitimate traffic used for modeling is readily available, traffic is hard to be collected due to the versatile VMs, virtual services, and network configuration. Moreover, since VMs use emulated CPU clock which is much less precise than the actual clock [13] [14], more noise exists in the network flows for such environments. Second, most intrusion detection systems (IDSs) are designed to detect TCs, whose intention is to transmit data faster with a higher bandwidth [6] [9] [11] [15]. Realizing this fact, the adversary, however, can design stealthy TCs, statistically indistinguishable from legitimate network flows, so as to deliberately evade detection.

In this paper, we propose a novel framework by taking the advantages of volatile VMs and dynamic configuration of SDN in a cloud infrastructure. Our framework can be dynamically enabled or disabled from the TC detection mode. Since our framework is particularly designed to detect TCs that perturb inter-packet delays (IPDs) of network flows, a new metric that performs differential analysis against outbound flows from different VMs is presented as well. In particular, the metric first decomposes the timing information of flows into different scales through discrete *wavelet-based multi-resolution transform* (DWMT). Then, a *Kullback-Leibler divergence*-based (KLD) metric is developed to measure the variance between flow pairs. The appealing feature of our approach is that it can detect most existing and stealthy TCs without legitimate traffic, even with the presence of noise and imprecise timekeeping mechanism in an SDN virtual environment. We implement our framework as a prototype system, OBSERVER, which can be dynamically deployed in an SDN cloud infrastructure. Our empirical experiments show that, compared with the existing TC detection approaches, OBSERVER can efficiently detect TCs with a higher detection rate, lower latency, and negligible performance overhead compared to existing methods. In summary, we have made the following contributions to this paper:

1) We have studied the challenges of detecting TCs in a networked virtual environment, particularly one in which legitimate traffic is unavailable and imprecise timekeeping mechanism is used. To counter the above challenges, we advance here a framework that leverages the spare VMs and the dynamic configuration of SDN to detect the TCs that perturb inter-packet delays (IPDs). The framework can be easily enabled or disabled via SDN-based instructions, replicate network packets, and receive timing statistics of outbound flows from VMs under comparison.

2) We have designed a metric that is both resilient to noise and sensitive to stealthy TCs. The metric is closely coupled with the proposed framework. It is a wavelet-based metric which can quantitatively calculate the distance among different outbound flows from VMs and thus detect a broad spectrum of timing channels.

3) We have implemented the novel detection framework, the metric, and empirically evaluated them against a number of TC attacks. To prove better efficiency, we compare our metric with other well-known metrics, which are designed for the physical network environment. Our evaluate shows that our approach can efficiently detect TCs with a higher detection rate, lower latency, and negligible performance overhead.

The remainder of this paper is structured as follows. Section 2 briefly introduces the background and the challenges of detecting TCs in an SDN virtual network. Section 3 reviews the related work. Section 4 presents the design of the TC detection framework. The key technical issues such as the threat model, definition and notations, and the detection metric are presented in this section. Section 5 introduces the design and implementation of our proposed framework. Section 6 presents the empirical evaluation results of detecting various TCs. Section 7 concludes the paper, addresses the possible improvements for OBSERVER, and suggests future research directions.

2. Background

In this section, we first introduce the threats of covert timing channel and the challenges involved to detect it in an SND virtual environment. Then, we present the effect of time drifting in a virtualized environment, which exacerbate the difficulties of anomaly detection.

2.1. Covert Timing Channel and Its Detection Challenges

A covert timing channel is a secrete communication channel that exfiltrates information from the compromised internal host to the external colluder and therefore violates the security policy [16]. There are two types of TCs: *active channel* (AC) and *passive channel* (PC). AC refers to the covert channel that generates additional traffic along with the existing traffic to transmit information, while PC refer to covert channel that manipulates the existing traffic and does not generate additional traffic [5]. In this paper, we only focus on the PCs that manipulate the inter-packet delays (IPDs) of a network flow [17]-[21]. The primary challenge of detecting TCs is that the statistics of TC flows are so close to those of legitimate flow that it is hard to detect them through traditional statistic tests. **Figure 1** illustrates the histogram and empirical cumulative distributions (ECD) of the inter-packet delays for a legitimate and a TC network flow sample, namely *JitterBug* [6] (sample size = 300). The distribution and ECDs of these two samples are very close. The statistics, such as means and standard deviations (or *stdev*), are also very similar as shown in **Table 1**. Low detection and high false positive rate are expected if one were to simply apply standard statistical-based detection metrics.

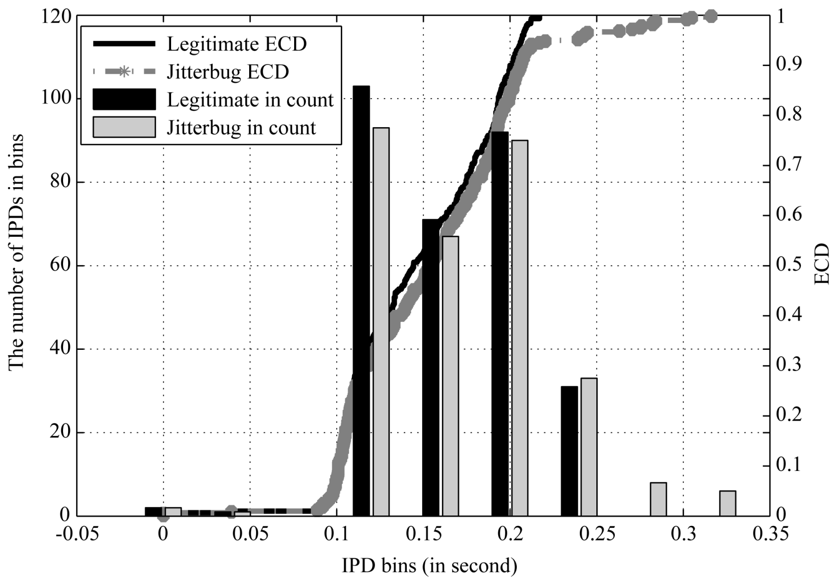

Figure 1. The comparison of ECD between a legitimate flow and a *JitterBug* flow.

Table 1. The means and standard deviations of two flows (in second).

Statistics	Legitimate flow	Jitterbug flow
Mean	0.14722	0.15097
Standard deviation	0.04064	0.04317

2.2. Time Drifting in Virtual Machines

Most existing TC detection approaches assume that the timekeeping mechanism in the hosts is accurate. This assumption might be true in a physical network environment, but might not hold in an SDN virtual environment for at least two reasons. First, unlike a physical machine that can directly access the physical CPU clock, a virtual machine accesses the CPU clock through either emulated timer devices or *virtual clock* [13], which makes accurate timekeeping impossible [14]. This fact affects all applications that access the virtual clock, including the malware that generates the TC. The noise introduced by the virtual clock, namely *time drifting*, has been mentioned in the literature [14]. **Figure 2** illustrates an observation of time drifting in the IPDs of outbound flows of two identical VMs when given the same inbound traffic. Even though the statistics of averaged IPDs of two outbound flows are very close, as shown in **Figure 2(a)**, time drifting can also be observed in a closer view in **Figure 2(b)**. Second, unlike a physical network environment, hosts and network connectivity in virtual networked environment are volatile [22]. For example, a VM may be arbitrarily migrated across virtual network or

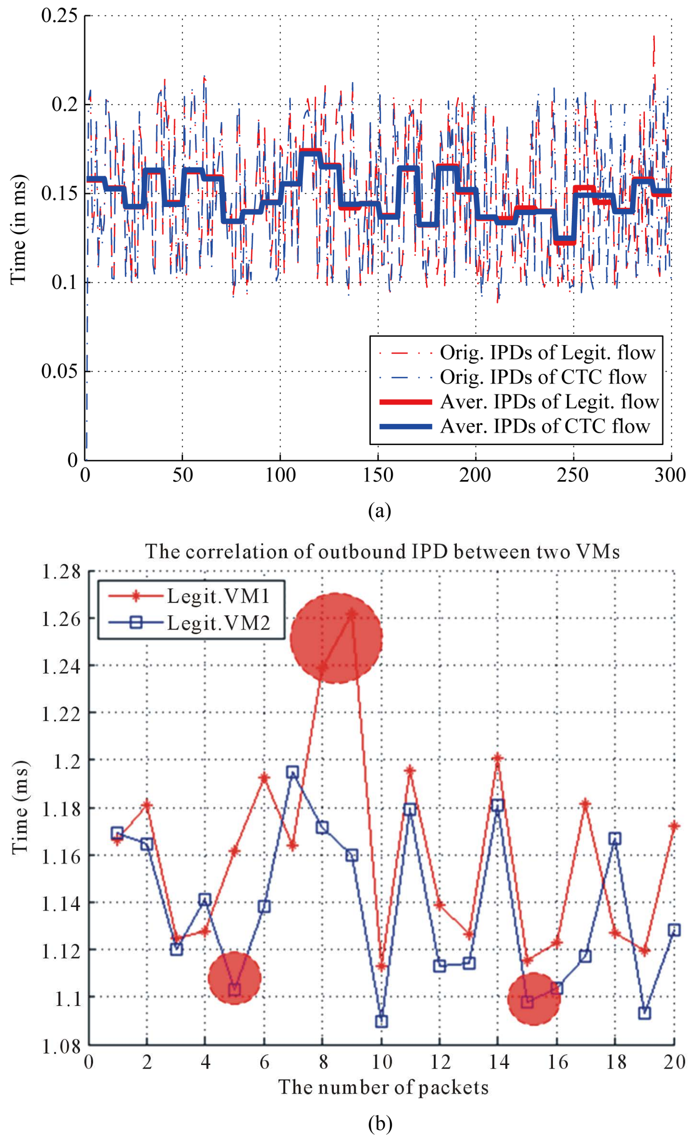

Figure 2. The effect of *time drifting* in virtual machines. (a) The similarity of IPDs between legitimate flow and TC flow (group size = 10); (b) A closer look at time drifting between two identical VMs without clock synchronization.

constantly reverted to a previous snapshot, which is a saved state of data and hardware configuration of a running virtual machine [23]. In addition, since a VM might be configured to run multi-booting systems, accessed by different users, and/or for different purposes, the network traffic might demonstrate different statistic patterns due to different "*pay-as-you-go*" services. Moreover, the topology of the SDN might be subject to change due to load-balancing or service migration [24]. Thus, it might be challenging to obtain legitimate traffic from the cloud provider. These factors cause legitimate traffic of a benign VM to reveal different statistics in time across its life-cycle, which makes the legitimate traffic for modeling useless. In either case, the previous anomaly detection approaches fail.

3. Related Work

The design of TCs has been the subject of recent research. For example, Berk *et al.* [10] implemented a simple binary covert timing channel based on the Arimoto-Blahut algorithm, which computes the input distribution that maximizes the channel capacity [25]. Cabuk *et al.* developed the first IP TC, which we refer to as IPTC [11] and TRTC [9], which is a more advanced traffic replay TC. Shah *et al.* [6] developed a keyboard device, called *JitterBug*, that slowly leaks user typed information over the Internet. Giffin *et al.* [26] showed that although not a TC, low-order bits of the TCP timestamp can be exploited to create a TC due to the shared statistical properties of timestamps and packet timing. Another application is to apply TC to trace suspicious traffic. For example, Wang *et al.* [17] took advantage of well-designed inter-packet delays, namely *watermark*, to trace VOIP traffic [18] [27]. They also utilize watermarked network traffic to trace-back bot-master through TC traffic [15]. Gianvecchio *et al.* [7] and Liu *et al.* [8] designed model-based covert channel encoding schemes that seek to achieve undetectability and robustness at the same time. Recent research of this field focuses on detecting covert channel in a multi-tenant virtualization environment, in which hostile tenants could leverage various covert channels to exfiltrate sensitive information from the victim, who shares the same physical machine [4] [28] [29]. While this line of research turns out to be relevant to ours, their primary focus is to design high bandwidth cross-VM covert channels, rather than detect them.

A number of TCs detection methods have also been developed. Peng *et al.* [30] showed that the *Kolmogorov-Smirnov* test is an effective way to detect TCs that manipulate IPDs. Cabuk *et al.* [9] investigated a regularity-based approach of detecting TCs. They also developed a metric, namely ϵ*-similarity*, to measure the proportion of similar inter-packet delays. The limitation of the ϵ*-similarity* metric is that it only targets to detect a particular TC, namely IPTC. Therefore, it is not general enough to detect other TCs. Berk *et al.* [10] employed a simple mean-max ratio test to detect binary or multi-symbol TCs. However, the mean-max ratio test assumes that the legitimate IPDs follow the normal distribution, which is often not true for real-world traffic. Gianvecchio *et al.* [5] investigated an entropy-based approach to detecting TCs, and they achieved good results of detection. All these detection schemes require legitimate network traffic to be available for modeling, and thus cannot be applied in a networked virtual environment, whose network traffic is volatile. Moreover, the model of legitimate network traffic has to be constructed off-line and incurs significant performance overhead, and thus makes real-time TC detection almost impossible.

Designed for a different purpose, Jing *et al.* proposed a wavelet-based approach to measure the time distortion of low-latency anonymous network [31] (or anonymity network [32]). Since their timing distortion metric is particularly designed to deal with the issues of timing distortion of IPDs, which are caused by anonymous network (e.g., Tor), such as flow mixing, merging, adding chaff, and packet dropping; it cannot be directly used to detect stealthy TCs, which demonstrate more subtle timing characteristics than the timing distortion of IPDs by anonymous network. Askarov *et al.* [33] [34] proposed online timing channels prevention mechanisms that mitigate information leakage. Although their approaches were proven to be efficient to identify the upper/lower bounds the information leakage as a function of elapsed time, how to detect timing channels is still an open problem that has not been addressed. The online prevention mechanisms without considering which network flow contains covert timing channel will significantly affect throughput and the response time of services, such as video streaming services which require a minimal transmission delay.

In our prior work [35] [36], we presented preliminary design for a TC detection system, which only considers TC detection in a simple networked virtual environment. Unlike existing approaches that take extra efforts to collect legitimate network traffic to construct the model, our approach did not require any legacy network traffic, but rather, the detection of TC relied on the differential analysis between multiple parallel outbound flows. Moreover, this approach has proven to be more efficient and robust in a networked virtual environment. This paper extends the preliminary results reported in our prior work as follows. First, we show that the system design given in [35] [36] lacks as it does not consider the timing to start and stop the TC detection system, which might waste the resources (e.g., the VMs) in a cloud infrastructure and could limit the applicability of its real-world deployment. Therefore, we not only address more realistic assumptions in the threat model, but also redesign our earlier system so that it is applicable in an SDN virtual environment. In our current design, the TC detection is initiated by the heuristic engine and terminated by the SDN controller, whose performance overhead is less than our previous implementation and easy to be automated. Furthermore, while our earlier system was designed and implemented in *C* and *divert socket*, our current design leverages the existing features of *OpenFlow* [37], which makes the TC detection system extremely easy to be set up, deployed, and upgraded. The SDN controller can enable and disable connections between virtual machines faster than our previous implementation and with minimal overhead. Second, to show the effectiveness of TC detection, a set of new experiments has been performed. For instance, we compare the true positive and false positive rates of our approach to those of existing TC detection approaches. Our results show that our approach outperforms existing approaches. Moreover, we also justify the choice of parameters for our WBD metric. Compared to our prior work, the current design and implementation of our TC detection system is not only readily deployable in a realistic SDN virtual network; but also much more efficient for TC detection. Finally, we discuss the limitations of our current approach, address the possible improvements, and suggest several future research directions.

4. Timing Channel Detection Framework

In this section, we first present the threat model of timing channel detection in an SND virtual environment. Then, we model our framework by providing notation and definitions. After that, we model the TC detection problem and formulate the metric that quantifies the divergence between network flows.

4.1. The Threat Model

In our threat model, we assume that the adversary launches insider attacks [38] [39] from within the well-protected enterprise, harvests secrete or confidential data from the compromised VMs, and transmits them to the outsider colluders. To maintain stealthy communication without triggering any security alert by the firewall or intrusion detection systems, the adversary can encode the data as passive timing channels (PTCs) that manipulate inter-packet delays (IPDs) between network packets. Since most anti-virus software, intrusion detection systems, and firewalls were not designed to detect TCs, they can easily be evaded in such a way that inside information could be exfiltrated. Although some malware and intrusion detection systems can be installed inside a VM, they can easily be turned off by the adversary, if the adversary gains the root privilege. Furthermore, since the VMs might be under the full control of the adversary, their malicious behavior can only be observed from the outside and should be therefore treated as black boxes.

We assume the cloud infrastructure reserves a large VM repository that handles a soaring number of user requests. This condition can easily be satisfied in an SDN or a Network Function Virtualization (NFV) [40] environment. An SDN controller can be configured to choose the VMs not in use or *vacant* VMs during detection and then revert to their original state after usage. Although it might be expensive to use vacant VMs during the detection process, we argue that it is feasible and practical to satisfy this requirement for at least two reasons. First, the TC detection system will only be enabled when it is needed: it will be enabled if an internal VM tries to establish a connection with an external unknown IP address or service and the existing IDS detect nothing malicious from the outbound traffic. It only samples a small amount of packets for analysis and raises the alert if any abnormality has been detected. Therefore, the usage of the vacant VMs is limited to a small percentage of the outbound traffic.

4.2. Notations and Definitions

Given the same network inbound flow I, the problem of measuring the timing distance between two outbound flows, namely O_1 and O_2, can be formulated as follows: The inbound flow I contains K packets $\left\langle p_{i,1},\cdots,p_{i,K}\right\rangle$ $(K > 0)$. In response to I, virtual machine VM1 generates O_1 that contains M packets $\left\langle p_{o1,1},\cdots,p_{o1,M}\right\rangle$ $(M > 1)$ and virtual machine VM2 generates O_2 that contains N packets $\left\langle p_{o2,1},\cdots,p_{o2,N}\right\rangle$ $(N > 1)$. Since the packets in

O_i (i =1, 2) were generated by VM$_i$ in response to I, we can segment the packets in I and O_i based on their request/response relationship. Specifically, for the j^{th} inbound segment $I^j = \langle p^j_{i,1}, \cdots, p^j_{i,m} \rangle$, its response outbound segment in O_i is defined as $O^j_i = \langle p^j_{oi,1}, \cdots, p^j_{oi,o} \rangle$. We use $t^j_{(oi,k)}$ to represent the timestamp of the k^{th} packet in the j^{th} segment of O_i. Since $K > M$ and $K > N$, we can further aggregate O^j_i into W aggregated segments ($l \le W$ and $W > 0$). Therefore, packets in $O_{i,l}$ are denoted as $\langle p_{oi,l,1}, \cdots, p_{oi,l,m} \rangle$ ($m > 1$). The time-stamps of the k^{th} packet in $O_{i,l}$ is denoted as $t_{(oi,l,k)}$. In **Figure 3**, O^{i-2}_2 is the response outbound segment of I^{i-2} in flow O_2. The segments O^{i-2}_2 and O^{i-1}_2 are aggregated as an aggregated one in O_2.

Definition 1. The *bit rate* $R(C_i)$ of a covert channel C_i is defined as the number of bits that are conveyed per unit of time. Given two covert channels, C_1 and C_2, if $R(C_i) < R(C_j)$, we say C_i is *stealthier* than C_j or C_j is more *aggressive* than C_i.

4.3. Measuring Timing Distance between Flows

By characterizing the timing patterns of the network flows, it is possible to quantitatively measure the timing distance between network flows. To effectively perform the measurement, we leverage discrete wavelet-based multi-resolution transform (DWMT) [41]. The DWMT, which has been widely used in signal processing [42] and anomaly detection [31], has at least three prominent features. First, DWMT provides multi-resolution analysis, which allows to look at the sequence of data at different scales. Second, DWMT allows feature localization, *i.e.*, it allows to know the characteristics of the signal and approximately where in time they occur. Finally, DWMT supports online analysis, that is, one can compare the difference between the two flows online.

The DWMT takes a sequence of data as input and transforms that sequence into a number of wavelet coefficients sequences. Specifically, the l level DWMT takes a sequence of IPDs and transforms the sequence into 1) I wavelet detailed coefficient vectors at different scales (CD_i, where $I \le i \le l$) and 2) a low-resolution approximate vector (CA_l)[1]. For the j^{th} segment of O_i, the wavelet detailed coefficients vector at scale I can be represented as:

$$V(i,j,l) = \left\langle CD^j_l(o_i,1), \cdots, CD^j_l(o_i,N_j) \right\rangle \tag{1}$$

where $N_j = n_j \times 2^{-j}$ is the number of wavelet coefficients at scale j, and $c_{l,k} = c_{l-1,2k} + c_{l-1,2k+1}$ for $l \ge 0$. **Figure 4** illustrates a particular DWMT, namely the *Harr wavelet* [43], on a IPDs sequence (sample size = 300). The Harr wavelet shown uses five resolution levels to yield one approximate coefficient vector a_5 and five detailed coefficient vectors d_i at different scales ($1 \le i \le 5$).

The design goals of our wavelet-based distance (WBD) are three-fold: First, we expect that the WBD between two legitimate flows is small. Second, we expect that the WBD between a legitimate flow and a TC flow is detectably different. Third, the WBD should be able to differentiate the more aggressive TC and the stealthier TC. To achieve these goals, we define three derived vectors based on the coefficient vector $V(i,j,l)$ at scale $l(l \ge 0)$: the *intra-flow vector* (intraFV), the *inter-flow vector* (interFV), and the *Kullback-Leibler divergence* (KLD) vector. We first define $\text{intra}(i,j,l) = \left\langle CD^j_l(O_i,1) - CD^j_l(O_i,0), \cdots, CD^j_l(O_i,N_j) - CD^j_l(O_i,N_{j-1}) \right\rangle$, which reflects the fluctuating characteristics between adjacent coefficients within a coefficient vector at scale j of one flow O_i. Then, we define $\text{intraFV}(j,l)$ as the Euclidean distance [44] *dist* between $\text{intra}(1,j,l)$ and $\text{intra}(2,j,l)$ for the j^{th} segment of O_1 and O_2, in Equation (2):

$$\text{intraFV}(j,l) = dist\left(\text{intra}(1,j,l), \text{intra}(2,j,l)\right) \tag{2}$$

Similarly, we define the $\text{interFV}(j,l) = dist\left(V(1,j,l), V(2,j,l)\right)$ as the Euclidean distance between coefficient vectors $V(1,j,l)$ and $V(2,j,l)$, which characterizes the deviation between two wavelet coefficients for the same segment j at the same scale j. The *Kullback-Leibler divergence* (KLD) has been used to measure the distance between two probability distributions $p_1(x)$ and $p_2(x)$ [45] [46]. From an information theory's perspective, KLD measures the expected number of extra bits required to code samples from $p_1(x)$ when using a code based on $p_2(x)$. For probability distributions $p_1(x)$ and $p_2(x)$, and the number of bins as $|x|$, the KLD of $p_2(x)$ from $p_1(x)$ is defined as follows:

$$\text{KLD}\left(p_1(x), p_2(x)\right) = \sum_{i=0}^{|x|} p_1(x) \log \frac{p_1(x)}{p_2(x)} \tag{3}$$

[1]In this paper, we interchangeably use CA_j and CD_0.

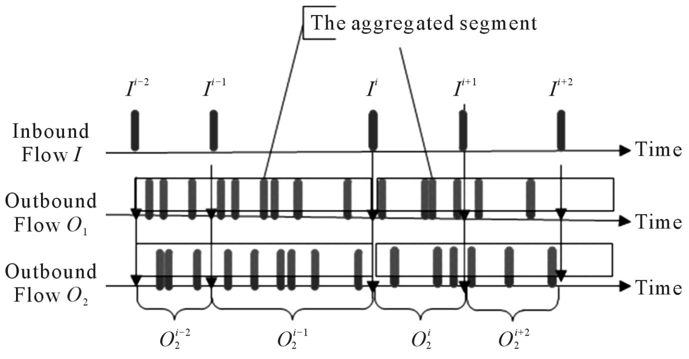

Figure 3. The inbound and outbound flows.

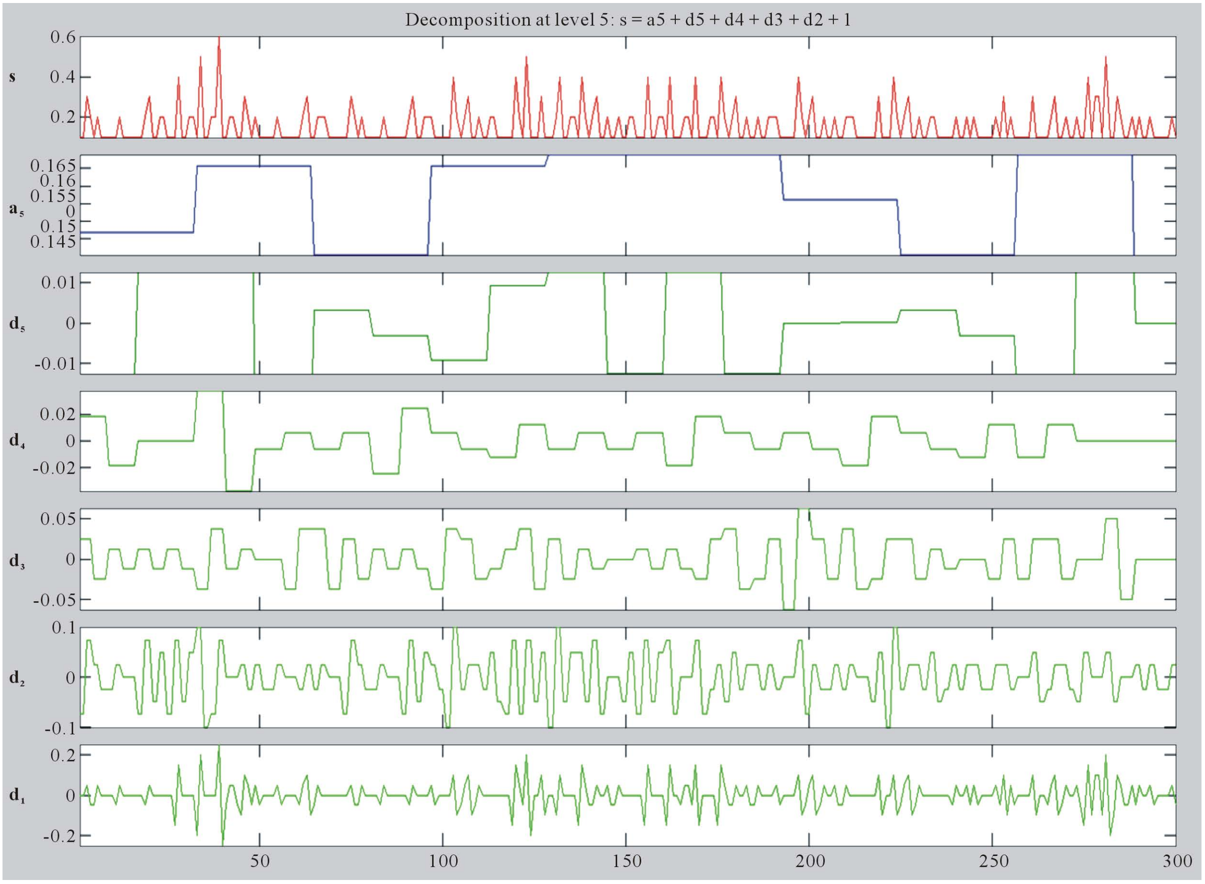

Figure 4. The original IPDs and its wavelet coefficients at different scales.

To calculate the KLD between two wavelet coefficient vectors at the j^{th} scale, it is necessary to obtain the probability distribution of $V(i,j,l)$, namely $p(V(1,j,l))$. To obtain $p(V(1,j,l))$, we first convert $V(i,j,l)=\left\langle CD_l^j(o_i,l),\cdots,CD_l^j(o_i,N_j)\right\rangle$, which contains numeric coefficients into a vector of symbols $\widetilde{S_\iota}=\langle\alpha_1,\cdots,\alpha_l\rangle$. Then, we calculate $p(V(i,j,l))$ based on $\widetilde{S_\iota}$. The effectiveness of converting from $V(i,j,l)$ to $\widetilde{S_\iota}$ depends on a mapping function F that maps $CD_l^j(o_i,m)$ into an alphabet $\mathcal{A}=\alpha_1,\cdots,\alpha_m$, where the soundness of translation from $CD_l^j(o_i,m)$ to α_m holds as follows:

$$F\left(CD_l^j(o_i,m)\right)=\alpha_m \text{ iff. } \beta_{j-1}\le\alpha_m\le\beta_{j-1}\,(1\le i\le l,1\le j\le m) \quad (4)$$

To facilitate effective conversion, we use the data discretization scheme of SAX [47] to assign wavelet coefficients into k equiprobable regions. Each continuous coefficient value that falls into a region maps to a unique symbol $\alpha_i\left(1\le i\le\kappa\right)$. **Table 2** illustrates the equiprobable regions as defined under $N\left(0,1\right)$ Gaussian distribution, in which the area between β_j and $\beta_{j+1}\left(1\le j\le 8\right)$ is $\frac{1}{\alpha}\left(3\le\alpha\le 10\right)$ [2]. In addition, since KLD is not symmetric: For two probability distributions $p_1\left(x\right)$ to $p_2\left(x\right)$, the KLD from $p_1\left(x\right)$ to $p_2\left(x\right)$ is generally not the same as the KLD from $p_2\left(x\right)$ to $p_1\left(x\right)$. Therefore, we defines the KLD for the j^{th} segment at scale l between $V\left(1,j,l\right)$ and $V\left(2,j,l\right)$ as follows:

$$\begin{aligned}\mathrm{KLD}\left(j,l\right)&=\frac{\mathrm{KLD}\left(p\left(V\left(1,j,l\right)\right),p\left(V\left(2,j,l\right)\right)\right)+\mathrm{KLD}\left(p\left(V\left(2,j,l\right)\right),p\left(V\left(1,j,l\right)\right)\right)}{2}\\&=\sum_{i=0}^{k}\frac{\left(p\left(\widetilde{S}_1\right)\log\frac{p\left(\widetilde{S}_1\right)}{p\left(\widetilde{S}_2\right)}+p\left(\widetilde{S}_2\right)\log\frac{p\left(\widetilde{S}_2\right)}{p\left(\widetilde{S}_1\right)}\right)}{2}\end{aligned}\tag{5}$$

There is a tradeoff involved in choosing the optimal size κ of alphabet A. That is, a large value κ keeps more information about the distribution of the continuous data, but it might generate too many false alarms. In contrast, a small value of κ keeps less information about the distribution, but it might be insensitive to the detection of slow or stealthy attacks. In Section 6.2, we will examine more closely this tradeoff and illustrate how we can determine the optimal value of κ.

Given the number of scale L and the number of aggregated segment W in a pair$\left\langle O_1,O_2\right\rangle$, we define the wavelet-based distance (WBD) between O_1 and O_2 as shown in Equation (7). WBD summarizes the divergence of inter-flow, inter-flow, and KLD between two network flows. A large value of WBD indicates significant difference between two flows, while a small value of WBD implies the opposite.

$$\mathrm{WBD}\left(O_1,O_2\right)=\sum_{j=1}^{W}\left(\sum_{l=0}^{L}\mathrm{intraFV}\left(j,l\right)\times\sum_{l-0}^{L}\mathrm{interFV}\left(j,l\right)\times\sum_{l-0}^{L}\mathrm{interFV}\left(j,l\right)\right)^2\tag{6}$$

We further extend the above definition to the normalized WBD as follows: given a number of outbound flows pairs$\left\langle O_1,O_2\right\rangle$ $\left(1\le i\le m,1\le j\le n\right)$, and one of which, $\left\langle O_{l1},O_{l2}\right\rangle$, comes from legitimate VMs; the *normalized* WBD (N_WBD) is defined as follows:

$$\mathrm{N\_WBD}\left(O_1,O_2\right)=\frac{\mathrm{WBD}\left(O_1,O_2\right)}{\mathrm{WBD}\left(O_{l1},O_{l2}\right)}\tag{7}$$

where the flow pair$\left\langle O_{l1},O_{l2}\right\rangle$ is called the *baseline* flow pair.

Table 2. The equiprobable regions as defined under $N(0, 1)$ Gaussian distribution.

β \ α	3	4	5	6	7	8	9	10
β_1	−0.43	−0.67	−0.84	−0.97	−1.07	−1.15	−1.22	−1.28
β_2	0.43	0	−0.25	−0.43	−0.57	−0.67	−0.76	−0.84
β_3	-	0.67	0.25	0	−0.18	−0.32	−0.43	−0.52
β_4	-	-	0.84	0.43	0.18	0	−0.14	−0.25
β_5	-	-	-	0.97	0.57	0.32	0.14	0
β_6	-	-	-	-	1.07	0.67	0.43	0.25
β_7	-	-	-	-	-	1.15	0.76	0.52
β_8	-	-	-	-	-	-	1.22	0.84
β_9	-	-	-	-	-	-	-	1.28

[2]Although the continuous wavelet coefficients may follow other distributions, this does not affect the effectiveness for calculating the equiprobable regions once the distributions have been identified.

5. System Design

In this section, we first present the architecture of our time channel detection system, which leverages the vacant VMs and the SND virtual network. Then, we detail the implementation of our TC detection system, as well as the evaluation tests using our proposed metric.

5.1. System Architecture

The architecture of OBSERVER is illustrated in **Figure 5**, which comprises four major components. Initially, the *heuristic engine* (HE) is triggered by a network packet sniffer (e.g. Wireshark [48]) or a network intrusion detection system (NIDS), when an internal VM (say, VM1) tries to establish a connection with an unknown server outside and the existing IDS cannot detect anything malicious from the outbound traffic. HE activates the OBSERVER in the TC detection mode. Specifically, the HE sends instructions to *the SDN controller* (SC) and *VM controller* (VC), respectively (Step 1). As a consequence, the SC issues the command that enables the virtual bridge (Step 2a). Meanwhile, the VC issues the command that starts VMs from the VM repository (Step 2b). In this example, VM2 and VM3 are the VMs started from the repository. SC also enables vSwithch2 and vSwithch3, which allow the inbound traffic to be replicated by vSwithch1 and forwarded to VM2 and VM2, respectively (Step 3). Once the outbound flows are captured by vSwitch$_i$ (i = 1, 2, and 3) (Step 4), the timing information of each flow is forwarded to the *TC Dete*ctor (TD), which applies our TC metric to identify anomaly outbound flows (Step 5). If no TC is detected from the VM under inspection, SC will disable the virtual bridge and VC will reclaim the VMs in use to the repository.

As a key component, TD uses timing distance metric to measure the distance between the outbound flows of VM1 and VM2 (denote as *dist* (VM1, VM2)) and that of VM2 and VM3 (denote as *dist* (VM2, VM3)), respectively. TC uses *dist* (VM1, VM2) as the baseline to tell if the outbound traffic of VM1 is anomaly as follows: First, the targeted false positive rate is set at σ. To achieve this false positive rate, the cutoff scores—the scores that decide whether a sample is legitimate or covert, are set at the $(1-\sigma)\times 100^{\text{th}}$ or $\sigma\times 100^{\text{th}}$ percentile (high scores or low scores for different tests) of legitimate traffic scores. Then, samples with scores worse than the cutoff are identified as TC, while sample with scores better than the cutoff are identified as legitimate. The false positive rate is the proportion of the legitimate traffic samples that are wrongly identified as TC, while the true positive rate is the proportion of TC samples that are correctly identified as covert.

5.2. System Implementation and Evaluation Tests

We have implemented our TC detection system in *C* on the top of *KVM* [49]. The traffic filter and traffic distributor were implemented as a transparent bridge by *Open vSwitch* [50]. To emulate the malicious program that

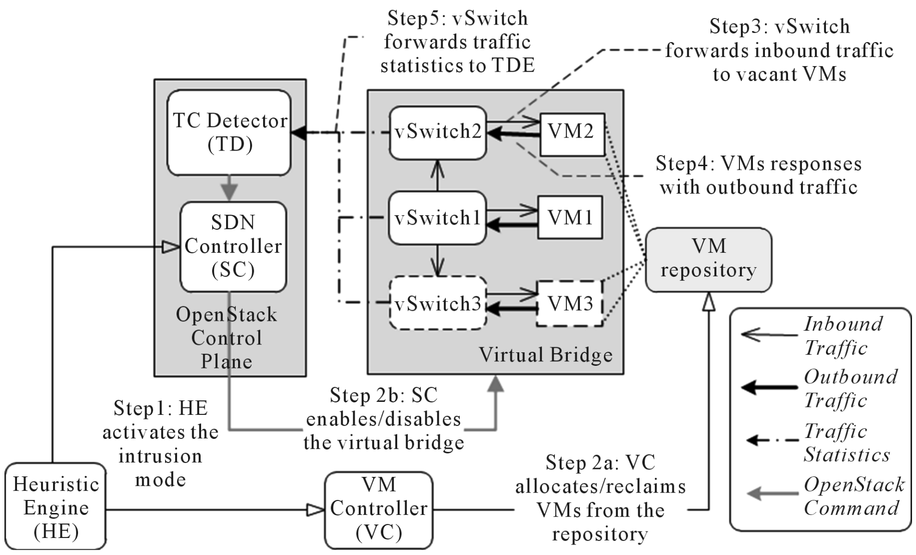
Step5: vSwitch forwards traffic statistics to TDE
Step3: vSwitch forwards inbound traffic to vacant VMs
TC Detector (TD)
SDN Controller (SC)
OpenStack Control Plane
vSwitch2
vSwitch1
vSwitch3
VM2
VM1
VM3
Virtual Bridge
Step4: VMs responses with outbound traffic
VM repository
Inbound Traffic
Outbound Traffic
Traffic Statistics
OpenStack Command
Step1: HE activates the intrusion mode
Step 2b: SC enables/disables the virtual bridge
Step 2a: VC allocates/reclaims VMs from the repository
Heuristic Engine (HE)
VM Controller (VC)

Figure 5. System architecture of OBSERVER.

exfiltrates inside information, we have modified the source code of *vsftpd* [51] running *on RealTime Application Interface* for Linux (RTAI) [52]. The modified *vsftpd* includes the TC encoder, which generates timing delay before sending outbound packets to the client. The TC encoder was written in *C* and inline assembly that invokes the *Read Timestamp Counter* (RDTSC) instructions of CPU [53]. We choose RDTSC instruction because it has excellent resolution and requires low overhead to generate timing delay [7].

In the following evaluation, we compare our *wavelet-based distance* (WBD) metrics and normalized WBD metrics with three other metrics to detect TCs: statistical tests, the timing distortion metric for measuring low-latency anonymous network [31], and the corrected conditional entropy method [5]. For the statistical tests, we use: 1) *shapetests*, which describes the first-order statistics, e.g., mean, standard deviation, and empirical cumulative distribution (ECD); 2) *Kolmogorov-Smirnov test* (KS-Test) [54]; 3) *Welch's T-test* (WT-Test) [55]; 4) *the regularity test* (RT-Test), which is used to determine whether the variance of the IPD is relatively constant or not [5] [7] [9] [11].

6. Evaluation

In this section, we first justify the assumption of the baseline selection by analyzing the similarity between legitimate outbound flows (Section 6.1). Then, we present the selection of the optimal values of parameters which might affect the effectiveness of detection (Section 6.2). After that, we compare our approach with existing approaches w.r.t. the effectiveness of detecting various TCs (Section 6.3).

6.1. Similarity between Virtual Machines for Legitimate Traffic

The objective of the first set of experiments is to justify our assumption that similar timing patterns exist between legitimate outbound flows as the baseline of TC detection. Given the same inbound traffic, the IPDs of outbound flows are collected from VM$_i$ (i = 1, 2, and 3). **Figure 6** illustrates the *quantile-quantile* (QQ) plot of

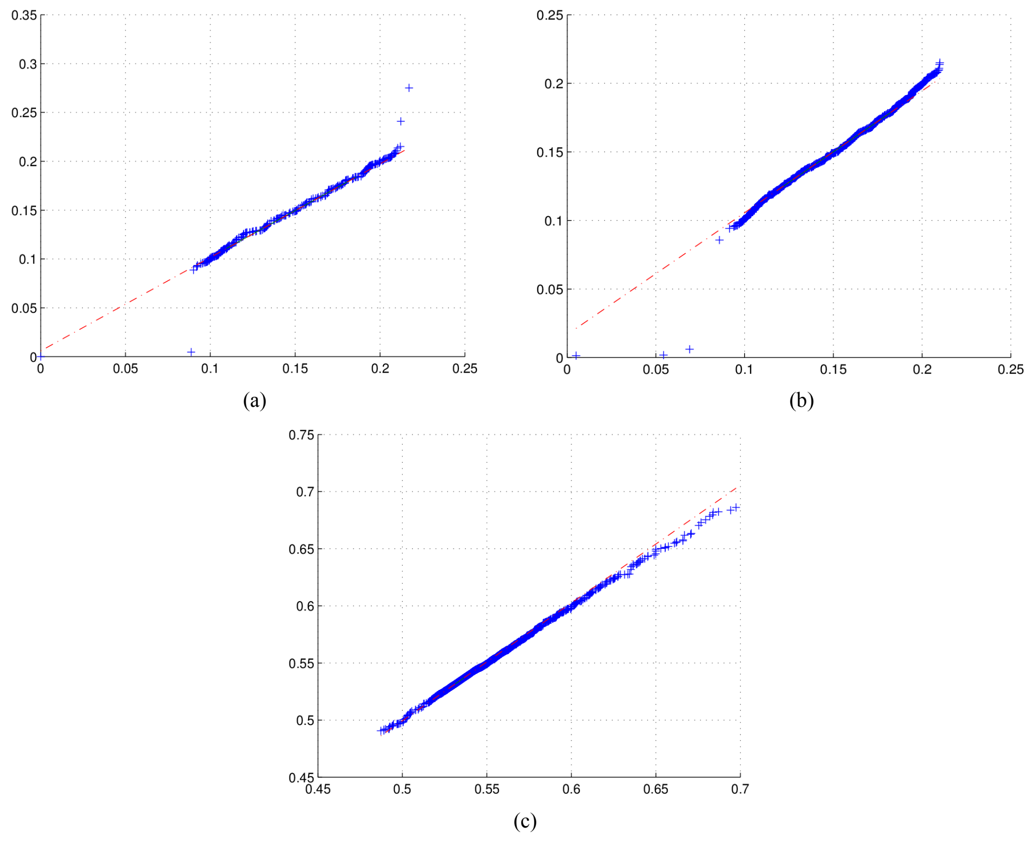

Figure 6. The *quantile-quantile* (QQ) plot of the IPD of two legitimate flows. (a) Sample size = 300; (b) Sample size = 1000; (c) Sample size = 33,000.

the IPDs of two legitimate outbound flows with the size 300, 1000, and 33,000. The highly linear nature of all plots strongly indicates that the outbound flows come from the same distribution.

Table 3 shows the test results from the datasets. For the WT-Test and the KS-Test at the 5% significance level, the value $h = 0$ indicates the acceptance of the null hypothesis; *i.e.*, that two flows come from the same distribution, while $h = 1$ indicates the opposite. The p-values of the WT-Test and the KS-Test indicates whether two samples differ significantly, say rejecting the null hypothesis if the *p-values* are "small". Both the WT-Test and the KS-Test accept the *null hypothesis* when the sample size is small (size = 100, 1000), but reject the hypothesis when the sample size is large (size = 33,000). However, our metrics show that all legitimate flows have constant values (less than 32.5), which indicating similar statistics between legitimate flows. In addition, this evaluation also indicates that more noise can be observed from the outbound IPDs if no long-term time synchronization mechanism is configured in VMs.

6.2. The Choice of κ and ω

In our detection approach, the *Kullback-Leibler divergence* that measures the difference between two discretized wavelet vectors one must choose κ, the size of the alphabet $\mathcal{A}$, which is critical to the effectiveness of detection. The tradeoff in choosing κ (see Section 4.3) are as follows: a smaller κ values keep less information about the individual distributionsleading to higher false negative rate. In contrast, a larger κ value increases the detection sensitivity, thereby leads to higher false positive rate. To determine the optimal value of κ in our system, we empirically test $\alpha = 2$ through 10 for both legitimate and different versions of Jitterbug flows. **Figure 7(a)** illustrates the WBD values of various flows, which indicates that legitimate and malicious flows can be differentiated when the value of κ is greater than 10. Weget a similar result in other TCs by running similar experiments. Thus, we choose $\kappa = 10$ to retain the ability of measuring the deviation of malicious traffic.

The second parameter to be selected is the detection window size ω, which is the number of IPDs to be compared between the outbound flows. Its selection also faces trade-off: the larger ω causes longer wavelet decomposition time and slower response, whereas the smaller ω cause the opposite and could make the detector over-sensitive and increase false positive rate. **Figure 7(b)** shows the WBD value for window size $\omega = 100$ through 1000 for both legitimate (e.g. normal) and stealthier Jitterbug flows (e.g. jitterbug_5, jitterbug_10, jitterbug_20, jitterbug_30). Our empirical results show that the WBD values of legitimate and malicious flows can be differentiated when ω is greater than 800. We obtain similar results for other TCs. Therefore, we choose $\omega = 1000$ for all the following evaluations.

6.3. Timing Channels under Evaluation

We evaluate our WBD metric with a wide range of TCs in **Table 4**, which covers both active TC and passive TCs. The empirical cumulative distribution (ECD) of IPDs for different TCs is illustrated in **Figure 8**. Below, we present the evaluation of detection for each of them in detail.

Table 3. The statistic values of IPDs of legitimate flows of different size.

Type of traffic	Data size = 300		Data size = 1000		Data size = 33,000	
	Legit1	Legit2	Legit1	Legit2	Legit1	Legit2
Mean	0.148	0.148		0.150	0.535	0.536
Standard deviation	0.039	0.040	0.035	0.033	0.009	0.009
WT-Test (h value)	0		0		1	
WT-Test (p value)	0.911		0.948		1.0E−003	
WT-Test (ci value)	[−0.007, 0.006]		[−0.003, 0.003]		[−0.537, −0.254]	
KS-Test (h value)	0		0		1	
KS-Test (p value)	0.986		0.282		6.57E−035	
KS-Test (ci value)	0.037		0.044		0.049	
WBD	32.525		5.3812		13.0492	

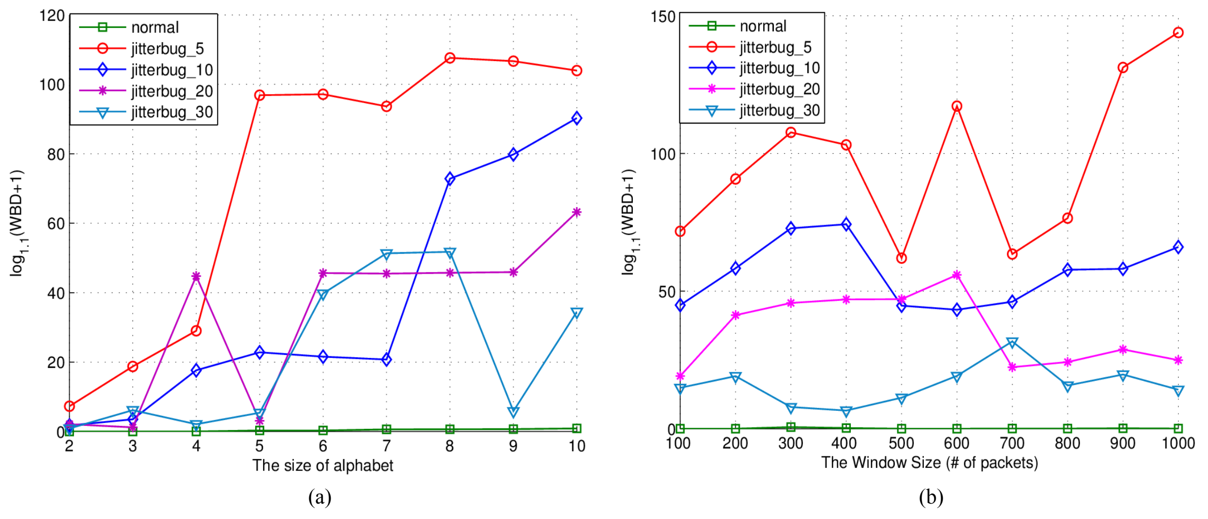

Figure 7. The WBD plots of different network flows for different κ and ω. (a) WBD plots of different network flows for different κ; (b) WBD plots of different network flows for different ω.

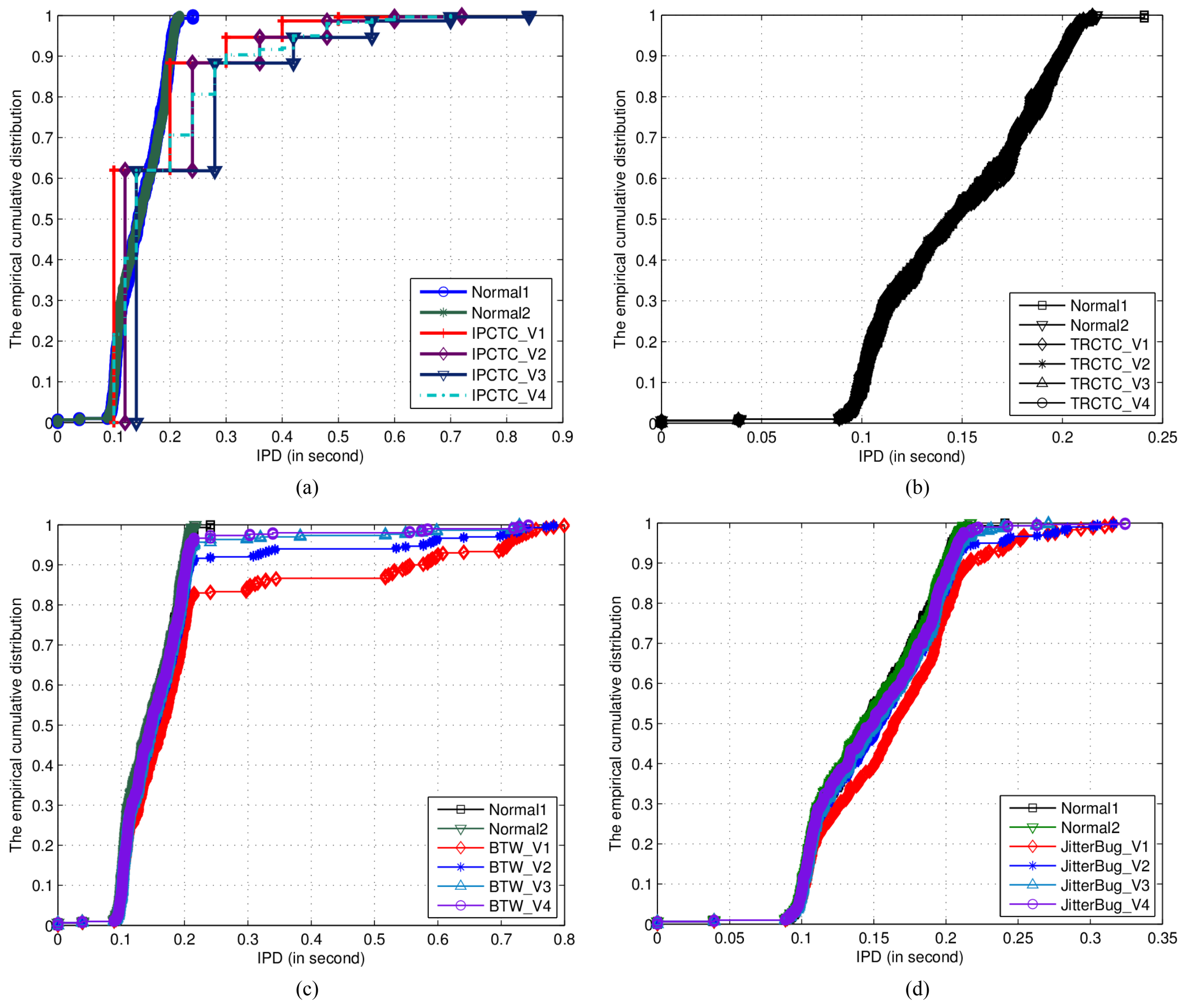

Figure 8. The *empirical cumulative distribution* (ECD) of IPDs for different TCs. (a) IPTC; (b) TRTC; (c) BTW; (d) *JitterBug*.

6.3.1. IP Timing Channel (IPTC) Detection

Our first sets of experiments are to detect the IPTC [11], whose working mechanism has been presented in Table 4. Specifically, IPTC encodes a 1-bit by transmitting a packet during a timing interval w, and encodes 0-bit by not transmitting a packet during w. In our experiment, the TC encoder reads the "*/etc/passwd*" file and encodes its binary into IPTC. We also developed four versions of IPTC, namely IPTC$_i$ ($i = 1, 2, 3$, and 4), whose design is the following: giving the observation that the average IPD of traffic is 0.147s, we design the first three schemes by choosing the timing interval w of 0.1s (IPTC1), 0.12s (IPTC2), 0.14s (IPTC3). The fourth scheme rotates among the above $w$after each 100 packets (IPTC4) to avoid creating a regular pattern of IPDs. In this experiment, we run the tests 100 times in a duration of 50 seconds. In each flow, we collect around 400 packets. In order to run the regularity test, we divide the IPDs sequence into 20 segments.

Figure 8(a) illustrates that even simple statistical tests can detect this attack: the empirical cumulative distribution (ECD) of legitimate flows and IPTC flows are different, whereas the ECD of two legitimate flows are similar. This situation matches with the findings in the literature [4] that IPTC is the easiest TC to be detected. Table 5 shows more detailed results of all tested flows. Although all the tests can detect all IPTCs, our WBD measurement shows the best results: the WBD of legitimate flows is very small (0.6295), which is in stark contrast to the WBDs of IPTC flows, all of which are 160,000 or higher. Although MLAN and CCE can differentiate all IPTCs from legitimate flow, they can hardly differentiate different versions of IPTCs. For instance, the CCE for different IPTCs are all close to the same value (0.609), and therefore indistinguishable. As an interesting observation, since IPTC$_i$ ($i = 1, 2, 3$, and 4) send packets at the same frequency, the regularities of them are identical. Thus, it is impossible to differentiate different IPTCs by using the regularity test. The WBDs values of IPTC$_i$ ($i = 1, 2, 3$, and 4), however, can be used to easily detect and differentiate different IPTC$_i$.

Table 4. Timing channels used in evaluation.

Name	TC description	Active/passive	Detectable?
IP Channel (IPTC) [11]	Transmitting 1-bit or 0-bit by choosing send or not send a packet during interval w	Passive	Yes
Time-Replay Channel (TRTC) [9]	Transmitting 1-bit or 0-bit by choosing historic legitimate IPDs as additional input from different baskets	Active	Yes
Botnet Traceback Watermark (BTW) [15]	Injecting modified control text	Passive	Yes
JitterBug [6]	Operates small delays in keystrokes to affect the original IPDs	Passive	Yes

Table 5. The test scores of legitimate and IPTC IPDs.

	Legitimate	IPTC$_1$	IPTC$_2$	IPTC$_3$	IPTC$_4$
Mean	0.1472	0.1560	0.1870	0.2182	0.1881
Standard deviation	0.0412	0.0893	0.1064	0.1242	0.1110
Regularity	0.972	0.9723	0.9723	0.9723	0.9723
WT-Test	0	1	1	1	1
KS-Test	0	1	1	1	1
KS-Test (p value)	0.967	0	0	0	0
MLAN	0.0433	1.3616	1.5072	1.7034	1.3530
CCE	1.1874	0.6055	0.6095	0.6099	0.7614
WBD	0.6295	15,982.6827	37,092.8784	76,188.4579	50,241.9389
N_WBD	1	25,389.488	58,924.3501	121,030.1158	79,812.4526

6.3.2. Time-Replay Timing Channels (TRTC) Detection

Our second set of experiments is to detect TRTC [9]. TRTC replays a set of legitimate inter-packet delays to mimic the legitimate traffic. Adversary first collects a sample of legitimate traffic, calculates their IPDs and put s them into a bin *Bin*. Then, *Bin* is partitioned into two bins of equal size: Bin_0 and Bin_1. TRTC transmits the bit 0 by randomly replaying an IPD from Bin_0 and transmits the bit 1 by randomly replaying an IPD from Bin_1. Since the IPDs in Bin_i (i = 0 and 1) are made up of legitimate traffic, the distribution of TRTC traffic is equal to that of the legitimate one. In this experiment, along with the original TRTC, four versions of TRTCs (TRTC1, TRTC2, TRTC3, TRTC4) are designed. In particular, one bogus packet is injected into the original flow after every n packets (n = 5, 10, 15, and 20), in which larger value of n indicates a slower attack. Figure 8(b) shows that the ECD of legitimate IPDs and four versions of TRTC IPDs are almost identical due to the encoding mechanism of TRTC.

Table 6 shows more detailed results of all the flows tested, which are similar to those derived for IPTC. Since TRTCs replays the legitimate IPDs, its IPDs demonstrate the same distribution as legitimate flows. Both WT-Test and KS-Test fail to detect TRTCs and thus accept the null hypothesis—the IPDs of TRTC and the legitimate traffic follow the same distribution. Although some tricky cutoff points can be chosen to detect some TRTCs through MLAN and CCE tests, the measurements of WBD and N_WBD of $TRTC_i$ are distinct enough such that they can easily differentiate various $TRTC_i$.

6.3.3. Back-Track Watermark (BTW) Detection

Back-track Watermark (BTW) is a passive TC specifically designed to track back the communication between a bot and its bot-master [15]. Specifically, to encode an i-bit sequence $S = s_0, \cdots, s_{i-1}$, the packets pairs: $\langle P_{ri}, P_{ei} \rangle$ $(i = 0, \cdots, L)$ are chosen randomly, such that $r_i \le e_i$, with P_{ri} called a *reference packet* and P_{ri} called an *encoding* packet. Let l_e and l_r be the IPDs of the covert bit encoding and reference packets, respectively. A covert bit $s_k\,(0 \le k \le i-1)$ is encoded into the packet pair $\langle P_{ri}, P_{ei} \rangle$. Specifically, the covert bit encoding function is defined in Equation (8). To adjust the intervals between P_{ei}, a pseudo-random number generator (PRNG) and seed s_t are used to generate the random time interval t_{ei} between P_{ei} and P_{ei+1}.

$$e(l_r, l_e, L, s_k) = l_e + \left[(0.5 + s_k)L - (l_e - l_r)\right] \bmod 2L \tag{8}$$

$$d(l_r, l_e, L) = \left\lfloor (l_e - l_r) \right\rfloor \bmod 2 \tag{9}$$

$$\text{If and only if } (-0.5 + 2i)L \le x_e - x_r \le (0.5 + 2i)L \tag{10}$$

We extend the initial TCs design scheme to generate slow attacks. In particular, we use $2ai\,(a \ge 1)$ packets to encode bit sequence *S*, where the parameter a can either be a constant or a variable generated by a PRNG. P_{ri} and P_{ei} were chosen from 2*a* packets. We call *a* the *amplifier*, which indicates how slow the information can be transmitted. The larger the value of *a*, the slower an attack can proceed. Four version of BTW have been developed by using different amplifiers.

Table 6. The test scores of legitimate and TRTC IPDs.

	Legitimate	$TRTC_1$	$TRTC_2$	$TRTC_3$	$TRTC_4$
Mean	0.1472	0.1465	0.1471	0.1466	0.1467
Standard deviation	0.0406	0.0411	0.0409	0.0410	0.0411
Regularity	0.2452	0.2958	0.1991	0.3009	0.1955
WT-Test	0	0	0	0	0
KS-Test	0	0	0	0	0
KS-Test (p value)	0.967	0	0	0	0
MLAN	0.0433	0.8700	1.0074	0.8995	0.8561
CCE	1.183	1.1589	1.1749	1.1813	1.1829
WBD	0.467	859.5832	651.9172	455.5041	318.9133
N_WBD	1	1840.6493	1395.9683	975.3835	682.8979

Figure 8(c) illustrates the ECD of two legitimate flows and four TC flows, which shows that both the legitimate and TC traffic follow the same distribution. More detailed results of all tested flows can be found in **Table 7**, in which the mean and standard deviations of legitimate and TC IPDs are quite close, particularly for large amplifiers. The WT-Test and KS-Test can detect aggressive BTW (BTW1) but all fail to detect stealthy BTW (BTW4). **Table 8** shows the comparison of true and false positive rate of different detection approaches, in which MLAN, CCE and WBD can reach 100% true positive rate. However, MLAN also suffers 65% false positive rate. Our evaluation also shows a tie between CCE and WBD as they both reach 100% true positive with zero false positive rate. However, a closer look at the quantities of CCE and WBD reveals that WBD gives better results than CCE because the WBD metrics are capable of differentiate the BTW_i, from the more aggressive to the stealthier ones.

6.3.4. *JitterBug* Detection

JitterBug [6] is a passive TC that manipulates the existing network traffic. In particular, it transmits a 1 bit by increasing an IPD to a value modulo w millisecond and transmits a 0 bit by increasing an IPD to a value modulo $\left\lfloor \frac{w}{2} \right\rfloor$ millisecond. For small values of w, the distribution of *JitterBug* traffic is very close to that of the original legitimate traffic. However, a too small w can also cause the TC flow indistinguishable from the legitimate flows with noises. In this experiment, we choose 100 milliseconds as the value of w. In addition, four versions of *JitterBug*, which use different amplifiers are also developed: *JitterBug*1 ($a = 5$), *JitterBug*2 ($a = 10$), *JitterBug*3 ($a = 20$), and *JitterBug*4 ($a = 30$). This design ensures that legitimate and TC flow have almost the same duration and comprise almost the same number of packets in flows. The ECD of Jitterbug is illustrated in **Figure 8(d)**.

Table 9 shows the scores of all tests, in which the means and standard deviations of legitimate flow (mean = 0.1472, stdev = 0.0406) and stealthy *JitterBug* flow, say *JitterBug*4 (mean = 0.1497, stdev = 0.0432), are very close. It is impractical to use regularity, WT-Test and KS-Test to detect *JitterBug* flows as they can only detect the regular *JitterBug* flows (JitterBug1) and all fail to detect the stealthy ones. Similarly, it is difficult to differentiate *JitterBugi* (i = 1, 2, 3, and 4) as the testing scores are the in a range between 1.18 and 1.20. In comparison, the testing scores of WBD can clearly distinguish *JitterBugs*, even the stealthiest one. The evaluation of true/false positive rates also demonstrates the advantage of our metric. **Table 10** tabulates the true/false positive

Table 7. The test scores of legitimate and BTW IPDs.

	Legitimate	BTW1	BTW2	BTW3	BTW4
Mean	0.1472	0.2207	0.1814	0.1627	0.1587
Standard deviation	0.0399	0.1775	0.1304	0.0918	0.0828
Regularity	0.2452	1.3504	0.7595	0.9706	0.9723
WT-Test	0	1	1	1	1
KS-Test	0	1	0	0	0
KS-Test (p value)	0.967	0.0004	0.2808	0.9862	0.9999
MLAN	0.0433	1.8296	1.2081		1.2394
CCE	1.1874	1.0440	1.0842	1.1079	1.1312
WBD	0.6295	1489.8973	828.7632	250.8589	124.9655
N WBD	1	556.8252	309.7370	93.7545	46.7039

Table 8. True positives (TPs) and false positives (FPs) of different detection approaches for BTW detection.

	NT-Test	KS-Test	MLAN ≥ 0.65593	CCE ≥ 0.9953	WBD ≥ 0.0015
Legitimate (FP)	0%	24%	65%	0%	0%
BTW1 (TP)	0%	87%	100%	100%	100%

Table 9. The test scores of legitimate and *JitterBug* IPDs.

	Legitimate	*JitterBug*1	*JitterBug*2	*JitterBug*3	*JitterBug*4
Mean	0.1472	0.1626	0.1549	0.1510	0.1497
Standard deviation	0.0406	0.0502	0.0482	0.0432	0.0432
Regularity	0.2452	0.2125	0.1848	0.2453	0.2969
WT-Test	0	1	0	0	0
KS-Test	0	1	0	0	0
KS-Test (p value)	1E−7	1E−7	1E−7	1E−7	1E−7
MLAN	0.0433	0.638	0.6957	0.8286	0.0701
CCE	1.1874	1.2012	1.2015	1.1933	1.1944
WBD	7.5561E−05	0.0218	0.0341	0.0349	0.0645
N_WBD	1	288.5086	451.2910	461.8785	853.6149

Table 10. True positives (TPs) and false positives (FPs) of different detection approaches for *JitterBug* detection.

	NT-Test	KS-Test	MLAN ≥ 0.60995	CCE ≥ 0.9953	WBD ≥ 0.00018
Legitimate (FP)	0%	31.4%	66%	5%	0%
BTW1 (TP)	62.4%	70.1%	100%	100%	100%

rates of different tests. Comparing to all other metrics, our metric is the only one that can achieve zero false positive rate and 100% true positive rate at the same time.

7. Conclusions and Future Directions

In this paper, we present a novel framework that addresses the open challenges of timing channel detection in an SDN cloud environment. Our framework dynamically configures SDN to enable/disable differential analysis against outbound network flows from different virtual machines (VMs). Coupled with a new metric, our framework is able to detect most existing and stealthy TCs without any traffic for modeling, even with the presence of noise and imprecise timekeeping mechanism. We implemented our framework as a prototype system, OBSERVER, which could be dynamically deployed in an SDN environment. Empirical evaluation shows that our approach can efficiently detect TCs with a higher detection rate, lower latency, and negligible performance overhead. This work was an initial exploration into the detection of network timing channels and there were many avenues for future work.

Future work will be aimed at addressing some of the shortcomings of the current design and implementation and extending the ideas explored in this paper. First, our current TC detection approach only focuses on detecting the active and passive TCs that manipulate the inter-packet delays (IPDs). However, in a multi-tenant virtualization environment, hostile tenants can leverage various covert channels to exfiltrate sensitive information from the victim, who shares the same physical machine [28] [29]. Covert channels were also observed in Android platform [56]. As a future direction, we plan to extend our approach to detect various TCs on different platforms, such as virtual and mobile platforms. Second, our wavelet-based distance metric can still be improved in many ways. In the short term, we plan to refine the time discretization algorithm using dimensionality reduction via PAA representation [57] and discretization based on different statistics [54]. In addition, we also plan to use better virtual clock synchronization techniques among VMs from the layer of Virtual Machine Manager (VMM) in order to increase the detection accuracy. Finally, although we have justified the usage of the vacant VMs in an SDN virtual network environment, this cost is still considered as inevitable overhead. An extreme case is that the adversary can trigger parallel outbound flows, among which only a negligible percent of them contain TC. This exploit could potentially impact the accessibility of the cloud by exhausting its resource. As another future direction, we will investigate other system architecture and detection metrics designed to be cost-efficient and robust in the face of attempts to use as little resources as possible. For example, instead of using the heavy-weight and high-cost vacant VMs, we will consider operating-system-level virtualization, or namely *container* [58]. The benefit of using container is that it imposes little to no overhead to the host machine. Moreover, the container can directly access the CPU clock on the host machine, which reduces the effect of time drifting in a virtual machine. As a summary, this work is an initial exploration into the detection of network timing channels and there are many avenues for future work. In the short term we will refine the detection metric by applying new time discretization algorithms. In the longer term, we will investigate the efficient system architecture and cost-efficient and robust detection metrics, which are capable of detecting various TCs in different platforms.

References

[1] Anonymous (2015) The Gray-World Team. http://gray-world.net/projects.shtml

[2] United States Government Accountability Office (2010) Cyberspace: United States Faces Challenges in Addressing Global Cybersecurity and Governance. Report to Congressional Requesters GAO-10-606.

[3] Salem, M.B., Hershkop, S. and Stolfo, S.J. (2008) A Survey of Insider Attack Detection Research. In: Stolfo, S.J., Ed., *Insider Attack and Cyber Security*: *Beyond the Hacker*, Springer, New York, 1-19.

[4] Ristenpart, T., Tromer, E., Shacham, H. and Savage, S. (2009) Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Cloud. *Proceedings of the* 16*th ACM Conference on Computer and Communications Security*, Chicago, 9-13 November 2009, 199-212. http://dx.doi.org/10.1145/1653662.1653687

[5] Gianvecchio, S. and Wang, H. (2007) Detecting Covert Timing Channels: An Entropy-Based Approach. *Proceedings of the* 14*th ACM Conference on Computer and Communications Security*, Alexandria, 29 October-2 November 2007, 211-230. http://dx.doi.org/10.1109/TDSC.2010.46

[6] Shah, G., Molina, A. and Blaze, M. (2006) Keyboards and Covert Channels. *Proceedings of the* 15*th USENIX Security Symposium*, Vancouver, 31 July-4 August 2006, 59-75.

[7] Gianvecchio, S., Wang, H., Wijesekera, D. and Jajodia, S. (2008) Model-Based Covert Timing Channels: Automated Modeling and Evasion. *Proceedings of the* 9*th International Symposium on Recent Advances in Intrusion Detection*, Cambridge, 15-17 September 2008, 211-230. http://dx.doi.org/10.1007/978-3-540-87403-4_12

[8] Liu, Y., Ghosal, D., Armknecht, F., Sadeghi, A.R., Schulz, S. and Katzenbeisser, S. (2009) Hide and Seek in Time: Robust Covert Timing Channels. *Proceedings of the* 14*th European Conference on Research in Computer Security*, Saint-Malo, 21-23 September 2009, 120-135. http://dx.doi.org/10.1007/978-3-642-04444-1_8

[9] Cabuk, S. (2006) Network Covert Channels: Design, Analysis, Detection, and Elimination. PhD thesis, Purdue University, West Lafayette.

[10] Berk, V., Giani, A. and Cybenko, G. (2005) Covert Channel Detection Using Process Query Systems. *Proceedings of FLOCON-CERT*, Pittsburgh, 20-22 September 2005.

[11] Cabuk, S. (2004) IP Covert Timing Channels: Design and Detection. *Proceedings of the* 11*th ACM Conference on Computer and Communications Security*, Washington DC, 25-29 October 2004, 178-187. http://dx.doi.org/10.1145/1030083.1030108

[12] Kreutz, D., Ramos, F.M.V., Verissimo, P., Rothenberg, C.E., Azodolmolky, S. and Uhlig, S. (2014) Software-Defined Networking: A Comprehensive Survey. *Computing Research Repository*, **103**, 14-76. http://dx.doi.org/10.1109/JPROC.2014.2371999

[13] Anonymous (2011) Timekeeping in VMware. http://www.vmware.com/files/pdf/Timekeeping-In-VirtualMachines.pdf

[14] Broomhead, T., Cremean, L., Ridoux, J. and Veitch, D. (2010) Virtualize Everything but Time. *Proceedings of the* 9*th USENIX Conference on Operating Systems Design and Implementation*, Vancouver, 4-6 October 2010, 451-464.

[15] Ramsbrock, D., Wang, X. and Jiang, X. (2008) A First Step towards Live Bot Master Trace Back. *Proceedings of the* 11*th International Symposium on Recent Advances in Intrusion Detection*, Cambridge, MA, 15-17 September 2008, 59-77. http://dx.doi.org/10.1007/978-3-540-87403-4_4

[16] Murdoch, S.J. (2008) Covert Channel Vulnerabilities in Anonymity Systems. PhD Thesis, University of Cambridge, Cambridge.

[17] Wang, X. and Reeves, D.S. (2011) Robust Correlation of Encrypted Attack Traffic through Stepping Stones by Flow Watermarking. *IEEE Transactions on Dependable and Secure Computing*, **8**, 434-449. http://dx.doi.org/10.1109/TDSC.2010.35

[18] Wang, X., Chen, S. and Jajodia, S. (2005) Tracking Anonymous Peer-to-Peer VoIP Calls on the Internet. *Proceedings of the* 12*th ACM Conference on Computer Communications Security*, Alexandria, 7-10 November 2005, 81-91. http://dx.doi.org/10.1145/1102120.1102133

[19] Kang M.H., Moskowitz I.S. and Lee D.C. (2007) A Network Version of the Pump. *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, 8-10 May 1995, 144-154. http://dx.doi.org/10.1109/SECPRI.1995.398929

[20] Wang, X. and Reeves, D.S. (2003) Robust Correlation of Encrypted Attack Traffic through Stepping Stones by Manipulation of Inter-packet Delays. *Proceedings of the* 10*th ACM Conference on Computer and Communications Security*, Washington DC, 27-30 October 2003, 20-29. http://dx.doi.org/10.1145/948109.948115

[21] Jia, W., Tso, F.P., Ling, Z., Fu, X., Xuan, D. and Yu, W. (2013) Blind Detection of Spread Spectrum Flow Watermarks. *International Journal of Security and Communication Networks* (*SCN*), **6**, 257-274. http://dx.doi.org/10.1002/sec.540

[22] Jansen, W. and Granc, T. (2011) SP 800-144: Guidelines on Security and Privacy in Public Cloud Computing. National Institute of Standards & Technology, Gaithersburg.

[23] fdsfadfad Anonymous. Hyper-V Virtual Machine Snapshots: FAQ. http://technet.microsoft.com/en-us/library/dd560637(WS.10).asp

[24] Wang, R., Butnariu, D. and Rexford, J. (2011) Open Flow-Based Server Load Balancing Gone Wild. *Proceedings of the* 11*th USENIX Conference on Hot Topics in Management of Internet*, *Cloud*, *and Enterprise Networks and Services*, Berkeley, 29 March 2011, 12-17.

[25] Blahut, R.E. (1972) Computation of Channel Capacity and Rate-Distortion Functions. *IEEE Transactions on Information Theory*, **18**, 460-473. http://dx.doi.org/10.1109/TIT.1972.1054855

[26] Giffin, J., Greenstadt, R., Litwack, P., and Tibbetts, R. (2002) Covert Messaging through TCP Timestamps. *Proceedings of the* 2*nd International Conference on Privacy Enhancing Technologies*, San Francisco, 14-15 April 2002, 194-208. http://dx.doi.org/10.1007/3-540-36467-6_15

[27] Wang, X., Chen, S. and Jajodia, S. (2007) Network Flow Watermarking Attack on Low-Latency Anonymous Communication Systems. *Proceedings of the* 2007 *IEEE Symposium on Security and Privacy*, Oakland, 20-23 May 2007, 116-130. http://dx.doi.org/10.1109/SP.2007.30

[28] Saltaformaggi, B., Xu, D., and Zhang, X. (2013) Busmonitor: A Hypervisor-Based Solution for Memory Bus Covert Channels. *Proceedings of the* 6*th European Workshop on Systems Security*, Prague, 14 April 2013, 1040-1042.

[29] Wu, Z., Xu, Z. and Wang, H. (2012) Whispers in the Hyper-Space: High-Speed Covert Channel Attacks in the Cloud. *Proceedings of the* 21*st USENIX Conference on Security Symposium*, Bellevue, 8-10 August 2012, 159-173. http://dx.doi.org/10.1109/TNET.2014.2304439

[30] Peng, P., Ning, P. and Reeves, D.S. (2006) On the Secrecy of Timing-Based Active Watermarking Trace-Back Techniques. *Proceedings of the* 27*th IEEE Symposium on Security and Privacy*, Oakland, 21-24 May 2006, 335-349. http://dx.doi.org/10.1109/SP.2006.28

[31] Jin, J. and Wang, X. (2009) On the Effectiveness of Low-Latency Anonymous Network in the Presence of Timing Attack. *Proceedings of the* 39*th Annual IEEE*/*IFIP International Conference on Dependable Systems and Networks*, Lisbon, 29 June-2 July 2009, 429-438. http://dx.doi.org/10.1109/DSN.2009.5270306

[32] Le Blond, S., Choffnes, D., Zhou, W., Druschel, P., Ballani, H. and Francis, P. (2013) Towards Efficient Traffic-Analysis Resistant Anonymity Networks. *Proceedings of the ACM SIGCOMM* 2013 *Conference on SIGCOMM*, Hong Kong, 12-16 August 2013, 303-314. http://dx.doi.org/10.1145/2486001.2486002

[33] Zhang, D., Askarov, A. and Myers, A. (2011) Predictive Mitigation of Timing Channels in Interactive Systems. *Proceedings of the* 18*th ACM Computer and Communication Security Conference*, Chicago, 17-21 October 2011, 563-574. http://dx.doi.org/10.1145/2046707.2046772

[34] Askarov, A., Zhang, D. and Myers, A. (2010) Predictive Black-Box Mitigation of Timing Channels. *Proceedings of the* 17*th ACM Computer and Communication Security Conference*, Chicago, 4-8 October 2010, 297-307. http://dx.doi.org/10.1145/1866307.1866341

[35] Liu, A., Chen, J.X. and Wechsler, H. (2013) Detecting Covert Timing Channels in a Networked Virtual Environment. *Proceedings of the* 9*th IFIP WG* 11.9 *International Conference on Digital Forensics*, Orlando, 28-30 January 2013, 273-288.

[36] Liu, A., Chen, J.X. and Yang, L. (2011) OBSERVER: An Real-Time System to Detect Covert Channels in a Highly Virtualized Environment. In: Butts, J. and Shenoi, S., Eds., *Critical Infrastructure Protection V*, Springer, Berlin, 151-164.

[37] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. and Turner, J. (2008) Open Flow: Enabling Innovation in Campus Networks. *Computer Communication Review*, **38**, 69-74. http://dx.doi.org/10.1145/1355734.1355746

[38] Shaw, E., Fischer, L. and Rose, A. (2009) Insider Risk Evaluation and Audit. Department of Defense Personnel Security Research Center, TR 09-02. http://www.dhra.mil/perserec/reports/tr09-02.pdf

[39] Cummings, A., Lewellen, T., McIntire, D., Moore, A. and Trzeciak, R. (2012) Insider Threat Study: Illicit Cyber Ac-

tivity Involving Fraud in the US Financial Services Sector. Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2012-SR-004. http://resources.sei.cmu.edu/asset_files/SpecialReport/2012_003_001_28137.pdf

[40] Sherry, J. and Ratnasamy, S. (2012) A Survey of Enterprise Middlebox Deployments. Technical Report UCB/EECS-2012-24, EECS Department, University of California, Berkeley.

[41] Addison, P.S. (2000) Multiresolution Signal Decomposition: Transforms, Subbands, Wavelets. 2nd Edition, Academic Press, Orlando.

[42] Addison, P.S. (2002) The Illustrated Wavelet Transform Handbook. CRC Press, Taylor & Francis Group, Boca Raton.

[43] Haar, A. (1910) Zur Theorie der orthogonalen Funktionensysteme. *Mathematische Annalen*, **69**, 331-371. http://dx.doi.org/10.1007/BF01456326

[44] Deza, E. and Deza, M.M. (2009) Encyclopedia of Distances. 2nd Edition, Springer, Berlin.

[45] Kullback, S. and Leibler, R. (1951) On Information and Sufficiency. *Annals of Mathematical Statistics*, **22**, 79-86. http://dx.doi.org/10.1214/aoms/1177729694

[46] Kullback, S. (1997) Information Theory and Statistics. Dover Publications, New York.

[47] Lin, J., Keogh, E., Wei, L. and Lonardi, S. (2007) Experiencing SAX: A Novel Symbolic Representation of Time Series. *Data Mining and Knowledge Discovery*, **15**, 107-144. http://dx.doi.org/10.1007/s10618-007-0064-z

[48] Anonymous (2015) Kernel Virtual Machine. http://www.linux-kvm.org/page/Main_Page

[49] Anonymous (2015) Wireshark, a Free and Open-Source Packet Analyzer. http://www.wireshark.org

[50] Anonymous (2015) Open vSwitch. http://openvswitch.org/download/

[51] Anonymous (2015) Vsftpd. https://security.appspot.com/vsftpd.html

[52] Anonymous (2015) Real Time Application Interface Official Website. http://www.rtai.org/index.php

[53] Anonymous (2015) Using the RDTSC Instruction for Performance Monitoring. http://www.ccsl.carleton.ca/~jamuir/rdtscpm1.pdf

[54] Hollander, M. and Wolfe, D.A. (1999) Nonparametric Statistical Methods. 2nd Edition, Wiley-Interscience, New York.

[55] Welch, B.L. (1938) The Significance of the Difference between Two Means When the Population Variance Are Unequal. *Biometrika*, **29**, 350-362. http://dx.doi.org/10.1093/biomet/29.3-4.350

[56] Gasior, W. and Yang, L. (2012) Exploring Covert Channel in Android Platform. *Proceedings of the* 2012 *International Conference on Cyber Security*, Washington DC, 26-28 Jun 2012, 173-177. http://dx.doi.org/10.1109/CyberSecurity.2012.29

[57] Karamitopoulos, L. and Evangelidis, G. (2009) A Dispersion-Based PAA Representation for Time Series. *Proceedings of the* 2009 *WRI World Congress on Computer Science and Information Engineering*, Los Angeles, March 31-April 2 2009, 490-494. http://dx.doi.org/10.1109/CSIE.2009.622

[58] Soltesz, S., Pötzl, H., Fiuczynski, M.E., Bavier, A. and Peterson, L. (2007) Container-Based Operating System Virtualization: A Scalable, High-Performance Alternative to Hypervisors. *SIGOPS Operating Systems Review*, **41**, 275-287. http://dx.doi.org/10.1145/1272998.1273025