Wang C, Min-Allah N, Guan B *et al.* An efficient approach for mitigating covert storage channel attacks in virtual machines by the anti-detection criterion. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 34(6): 1351–1365 Nov. 2019. DOI 10.1007/s11390-019-1979-8

An Efficient Approach for Mitigating Covert Storage Channel Attacks in Virtual Machines by the Anti-Detection Criterion

Chong Wang[1,2], *Student Member, CCF*, Nasro Min-Allah[3], Bei Guan[1], Yu-Qi Lin[4]
Jing-Zheng Wu[1], *Member, CCF*, and Yong-Ji Wang[1,2], *Senior Member, CCF*

[1] *Institute of Software, Chinese Academy of Sciences, Beijing 100190, China*
[2] *University of Chinese Academy of Sciences, Beijing 100049, China*
[3] *College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, Dammam 1982 Saudi Arabia*
[4] *Block Chain Research Center, Blue Helix, Grand Cayman KY1-1100, Cayman Islands*

E-mail: wangchong@nfs.iscas.ac.cn; nabdullatief@iau.edu.sa; guanbei@iscas.ac.cn; yuqi@nfs.iscas.ac.cn
jingzheng08@iscas.ac.cn; ywang@itechs.iscas.ac.cn

Received November 29, 2018; revised September 9, 2019.

Abstract Covert channels have been an effective means for leaking confidential information across security domains and numerous studies are available on typical covert channels attacks and defenses. Existing covert channel threat restriction solutions are based on the threat estimation criteria of covert channels such as capacity, accuracy, and short messages which are effective in evaluating the information transmission ability of a covert (storage) channel. However, these criteria cannot comprehensively reflect the key factors in the communication process such as shared resources and synchronization and therefore are unable to evaluate covertness and complexity of increasingly upgraded covert storage channels. As a solution, the anti-detection criterion was introduced to eliminate these limitations of cover channels. Though effective, most threat restriction techniques inevitably incur high performance overhead and hence become impractical. In this work, we avoid such overheads and present a restriction algorithm based on the anti-detection criterion to restrict threats that are associated with covert storage channels in virtual machines while maintaining the resource efficiency of the systems. Experimental evaluation shows that our proposed solution is able to counter covert storage channel attacks in an effective manner. Compared with Pump, a well-known traditional restriction algorithm used in practical systems, our solution significantly reduces the system overhead.

Keywords covert storage channel, information security, covert channel threat evaluation, security and privacy protection

1 Introduction

Lampson introduced the concept of covert channel (CC) in 1973 as the ability of high security level processes (HSLPs) to signal information to low security level processes (LSLPs)[1]. Since then, various extensions have been made and numerous studies have demonstrated that the covert channel is a real threat to information systems[2]. Existing literature suggests that covert channels can be broadly divided into two classes: 1) covert storage channel (CSC), and 2) covert timing channel (CTC). The former class encodes covert information by directly modifying attributes of shared resources (e.g., values of shared variables in an operating system)[3], while the latter class encodes covert information by manipulating transmission characteristics of shared resources (e.g., inter-packet delay)[4]. With the emergence of cloud computing[5,6] and virtual machine (VM)[7,8], there are even more opportunities to be exploited. Under such computing environments, resources such as caches and system variables are shared extensively within thin boundaries that can be compromised with covert channels[9−11].

Regular Paper
The work was supported by the National Natural Science Foundation of China under Grant No. 61772507 and the National Key Research and Development Program of China under Grant No. 2017YFB1002300.

VM covert channel is an effective mechanism for information exchange between two entities (the sender and the receiver) in cloud domain. On the other hand, these channels can also be exploited to launch malicious attacks such as leaking confidential information without being monitored. Among existing literature on VM covert channels[12,13], inter-VM covert channels[14] have received much attention. Under such arrangements, the sender and the receiver are placed in different VMs. The physical co-residency feature allows the adversary VM to communicate with the target VM on the same physical machine and leak the confidential information due to inappropriate logical isolation. Various VM provisioning strategies have been presented recently[12,15] to mitigate inter-VM channels. Another type of VM covert channels, namely intra-VM covert channel, can also result in confidential information leakage (e.g., kernel information and memory mapping information) due to adversaries' nonadherence with the counterpart system's security policies.

In cloud computing, intra-VM covert channels exist in a virtual machine that is installed on top of a modern operating system. While running, a virtual machine can grant permission to an LSLP (the receiver) for network access, while an HSLP (the sender) is restricted/denied from doing so. In this way, the HSLP has the access to confidential information, but cannot successfully leak it to LSLP by legitimate channels due to the restriction of the system's security policies. However, the HSLP can tend to resort to covert storage channels, so that it can subtly succeed in transmitting confidential messages to the LSLP. Subsequently, the LSLP spreads the information through the network. Combined with other types of attacks[2,16,17], intra-VM covert channels possessing confidential information are very likely to cause severe damages by using the leaked confidential information. In contrast to the inter-VM covert channel, the intra-VM covert channel is a process collusion attack and hence both the sender and the receiver of an intra-VM covert channel reside in the same VM. Therefore, the VM provisioning technique, which is essentially a scheduling algorithm and concentrates on multiple VMs, will lose its effect.

Many researchers have demonstrated that covert channels cannot be completely eliminated[2,18,19]. A possible solution for covert channel elimination in a virtualization scenario would be the separation of storage, which means every user or process owns their physically separated storage. However, such scenarios are costly and even impractical in many cases. As an alternative, one feasible solution is covert channel restriction. Unfortunately, restricting covert channels is often problematic as it can slow down system mechanisms or introduce noises. Both aforementioned cases limit the performance of computer systems[2,16,17]. Existing solutions are pessimistic in the sense that such techniques either compromise system flexibility or offer performance with higher vulnerability to be exploited. This paper presents a restriction algorithm called Virtual Machine Covert Storage Channel Restriction (VMCSCR) for intra-VM storage channels to achieve an appropriate trade-off between the system security and the system performance.

The principle of covert channel threat estimation is an essential part of VMCSCR. Specifically, the results of threat estimation provide a basis for VMCSCR to restrict each covert channel. The existing threat estimation criteria mainly include channel capacity, accuracy, and short message. Unfortunately, these criteria are only a description of results of the transmission process and cannot truly reflect the covertness and key elements of covert channel communication, i.e., shared resources, encodings, and synchronization mechanisms. Many researches have shown that covertness[20−22] is an indispensable criterion required for estimating threats of covert channels, because if a covert channel is detected, the channel is very likely to be restricted or eliminated by the system's security enforcer, and hence will finally lose effects regardless of the high capacity or accuracy associated with the channel. Considering such a scenario, this paper presents an anti-detection criterion (ADC) as a supplement to the current threat estimation criteria, which includes key elements of covert storage channel communication process in the formal definitions. ADC can quantify the covertness of covert storage channels and therefore provides the basis for VMCSCR.

In summary, this paper makes the following contributions as follows.

1) This paper presents the limitations of current threat estimation criteria and exploits the anti-detection criterion to eliminate such limitations.

2) This paper presents a covert channel restriction algorithm (VMCSCR) which is based on the anti-detection criterion and can be applied in scenarios where multiple covert channels exist simultaneously. We show that VMCSCR achieves a proper trade-off between the threat restriction and system performance overheads.

3) This paper evaluates the efficiency and efficacy

of VMCSCR in the VM environment. Experimental results show that VMCSCR provides guaranteed protection against information leakage via intra-VM covert storage channels.

2 Background

Covert channels can be implemented in varieties of ways according to the underlying systems. For instance, in operating systems and databases, the covert channel communication can be established between high and low security level processes. Realization of such a system is subject to the modification and perception of shared resources. In these systems, attackers implement covert channels to pass and leak systems' confidential information[2]. In network systems, an intruder uses communication protocols such as VoIP[23] and modulates covert information into inter-packet delays (IPDs) of a single packet stream. Thus, the attacker can bypass the firewall and intrusion detection system (IDS) to achieve the purpose of information leakage[24].

Due to the nature of virtual machines, covert channels pose a substantial threat to the isolation that exists among different security-level processes or different VMs. These processes or VMs extensively use shared hardware resources at different security levels and hence can disclose confidential information. CPU branch prediction table, instruction cache, and memory are among the possible hardware resources to be exploited. In data centers[25], the shared network resources can be utilized to transfer data between logically isolated virtual networks[26]. Other possible scenarios to build a covert channel include time, frequency and other resources[9,27,28]. Authors in [13] pointed out that even in a formally verified microkernel, covert channels can still be considered as a potential threat to information leakage.

As per requirements of CC (Common Criteria for Information Technology Security Evaluation), TCSEC (Trusted Computer System Evaluation Criteria), and current researches[2,29], covert channel analysis mainly includes the following four aspects: covert channel identification, construction, threat estimation, and restriction/mitigation. Through comprehensive analysis and scanning of the system, identification uses analytic methods[30−33] to discover the existing covert channels and show potential shared resources that can be used to construct covert channels. Construction is a way to build real-world covert communication scenarios and to simulate the attacker's behaviors. There exist two kinds of covert channel construction methods, namely the model-based construction method[9] and the protocol-based construction method[34−36].

3 Related Work

3.1 Threat Estimation

Threat estimation evaluates the potential threat of covert channels and provides guidelines and basis for designing channel restriction strategies. The current threat estimation criteria mainly include channel capacity, accuracy, and short message[9,34,37,38]. As a necessary supplement to the current criteria, the anti-detection criterion[39] is demonstrated in Section 5. We discuss related work according to the current threat estimation criteria as follows.

1) *Channel Capacity.* Channel capacity is the maximum transmission rate of covert channels and is generally adopted as the main criterion for threat estimation. Millen[34] was the first one to present finite state machines based on Shannon theory to model covert channels. Tsai and Gligor[40] replaced the finite-state automata with a Markov model to simulate covert channels and to estimate the channel capacity, which is widely used in practice. The methods presented by Millen[34] and Tasi and Gligor[40] have been adopted by TCSEC. Wu *et al.*[41] proposed an error-corrected four-state machine to estimate channel capacity and to deal with transmission errors. Lin *et al.*[9] used HLPN to estimate channel capacity.

2) *Small Message.* There are situations, where short messages are inevitable. For instance, some confidential information such as account and password is short but valuable. Unfortunately, the channel capacity is not suitable for describing such short messages. Moskowitz and Kang presented a small message criterion as a necessary supplement to channel capacity[18] that contains the length and fidelity of small messages, and the time of small messages transmission is also included. But this criterion cannot reflect the sensitivity of covert information. Therefore, Zeng *et al.* proposed short message transmission value[37] by redefining the small message criterion and successfully applied the concept to database systems.

3) *Accuracy.* Accuracy describes the fidelity of confidential information during a covert channel transmission and is measured as the percentage of correctly received bits. Binary-coded information is usually used for covert communication; therefore it is meaningless

to simply record how many bits are correctly transmitted. Cabuk *et al.* presented edit distance[24] to quantify the accuracy of a covert information transmission, where edit distance is the minimum distance between two strings[42].

Our previous work, which belongs to the field of covert channel threat estimation, presented the undetectability criterion to evaluate the threat of covert channels in traditional operating systems[39]. According to the requirements of covert channel analysis, the result of covert channel threat estimation can provide guidelines for threat restriction, and the undetectability criterion[39] can quantify the threat of a covert channel. In this paper, we improve the criterion and adopt it as the input for our restriction algorithm VMCSCR. The criterion is improved by adding new key factors to the criterion, and some parts of the criterion are redefined. Furthermore, the quantification and computing methods of the previous criterion are complex and are hard to establish. We redesign the computing method to form a simplified and specific calculation method. We also improve the criterion based on the characteristic of the virtualization and experiments.

3.2 Threat Restriction

Covert channel restriction is to mitigate or eliminate the threat of the channel completely based on threat estimation. Typical covert channel restriction technologies include addition of noise (or delay) to channels, interfering with the channel transmission process, and reducing the channel capacity (or accuracy)[41,43–45].

Hunger *et al.*[46] presented complementary guest-VM level software techniques to defend microarchitectural covert channels. Zhang *et al.*[47] used bystander workloads to mitigate the risk of cross-VM covert channels in virtual machines, which are usually adopted by a public cloud. The continuous time Markov process was used to model the impact of bystanders on the cross-VM covert channel in terms of both the work scheduling of the virtualization platform and the intensity of the bystander workloads. Caviglione *et al.*[48] presented a covert channel attack against personal cloud storage services. This kind of covert channel stealthy exchanges the covert information through the Internet. The performance of the presented covert channels is evaluated with Dropbox[1] in a production quality deployment.

Gai *et al.*[49] presented a model named PBEM-SGN (Permissioned Blockchain Edge Model for Smart Grid Network) for privacy protections and energy security. In this model, covert channel authorization techniques are adopted to guarantee users' validity. Evtyushkin *et al.*[50] restricted threats of contention-based covert channels through dynamic branch prediction units in modern processors. StealthMem proposes the use of page coloring to assign a confidential VM's physical memory pages to specific cache lines and then lock the lines in the cache[51]. Zhang *et al.* presented HomeAlone[52] to deal with threats of cache-based covert channels. HomeAlone reserves a dedicated machine and attempts to ensure that no other malicious VM is co-resident on the machine. Our work mainly belongs to covert channel restriction and is detailed in Section 6.

4 Assumption, Threat Model, and Motivation

4.1 Assumption

To understand our assumptions well, we consider a scenario where two aggressive malwares or compromised applications are running in a VM, namely a spy and a Trojan. Assuming that the Trojan is HSLP, and the spy is LSLP. As mentioned before, HSLPs have access to confidential information. Due to the system security policy, such information cannot be passed to LSLPs through legitimate channels. Eventually, covert channels are utilized for information transmission. For example, the Trojan can be a password manager while the spy can be a weather widget. Initially, the system is uncompromised so that it properly enforces security policies and access control and preserves legitimate information flows.

Due to the system's security policies, no other legitimate channel can exist between the Trojan and the spy. Therefore, covert channel is the only possible mechanism for them to communicate with each other. In a covert channel, let the Trojan be the sender while the spy is the receiver. Moreover, if the network access is granted, the spy has the potential to transmit sensitive information to a remote receiver. To highlight the significance of such channels, we consider a spy with network access in our experimentation.

4.2 Threat Model

A channel consists of five elements: the sender, the receiver, synchronization, encoding, and shared resources. The sender encodes the covert information into binary bits by changing the properties of shared

[1]www.dropbox.com, Sept. 2019.

resources. The receiver observes the changes and decodes the information accordingly. The synchronization and encoding ensure reliable communication between the sender and the receiver. Covert channels in communication scenarios may vary according to the underlying shared resources.

Previous researches[9,36], which belong to the field of covert channel construction, proposed three real-world covert communication protocols namely BP, SAP, and TCTP to construct practical covert channel scenarios. This paper adopts and slightly modifies BP and TCTP to illustrate the limitations of the current threat estimation criteria. Specifically, two modified *last_pid* channels[9,36] are used to steal root password and leak sensitive information in a Linux platform. These two channels utilize *last_pid* and temporary files as shared resources but with different synchronization respectively. *last_pid* denotes the maximum process ID allocated to a process. Under such scenarios, when a new process is forked, *last_pid* is incremented. All processes in different security levels share the value of *last_pid* irrespective of how system security policy is enforced. Temporary files are available in temporary folders, which are shared by all users. Although some Linux distributions limit the access to the content of temporary files, users can still check whether the temporary file exists or not by creating a new file.

Protocol 1. Protocol 1 used in this work is the same as the TCTP protocol implemented in [9, 36]. The sender and the receiver of protocol 1 are two processes running at different security levels in Linux. Protocol 1 uses temporary files to synchronize information and adopts shared resource *last_pid* to transmit.

Protocol 2. The only difference between protocols 1 and 2 is that instead of the temporary files, protocol 2 uses time T for synchronization. During any time T, if a bit to be sent is binary 0, the sender does not perform any operation on the shared resource. However, if a binary 1 is intended to be sent, the sender adds 2 to *last_pid*. On the receiving side, the receiver observes the value of *last_pid* in T. If its value remains unchanged, the receiver records bit 0, or records bit 1 if the value is incremented by 2.

4.3 Motivation

To understand our main results, we provide a brief sketch of the limitations of existing criteria pertaining threat estimations. Table 1 shows the ability of each criterion to estimate threats of the two protocols discussed in Subsection 4.2. It can be noticed that the shared resources and encoding scheme of the two protocols are the same. The only difference between them is the synchronization mechanism. The main factor affecting the synchronization of protocol 1 is the creation of the temporary files, but the time to create a file is basically fixed. The time T is the main factor that affects the synchronization mechanism of protocol 2. The channel capacity of protocol 2 can be intentionally adjusted by changing the value of T. Additionally, the accuracy of protocol 2 is also affected by T and shared resources. Table 1 suggests that when T is too large, it is very likely that the value of *last_pid* has been modified by other processes. For instance, the modification of *last_pid* may occur twice before the receiving peer is notified. On the other hand, when T is too small, the value of *last_pid* is possibly changed due to the overlapping modifications on *last_pid*. As we know, the small message criterion is similar to the channel capacity. If we only use capacity, accuracy, or small criterion as a threat estimation criterion, the threat of protocol 2 can be greater than, equal to, or inferior to protocol 1 by simply changing T. Thus, it is nearly impossible to get a specific threat estimation results based on current criteria.

Table 1. Differences Between the Current Criteria and the Anti-Detection Criterion When Evaluating Protocols 1 and 2

Criterion	Distinguishability	Main Reason
Capacity	×	Lack of synchronization
Small message	×	Lack of synchronization
Accuracy	×	Lack of synchronization
Anti-detection	✓	Including synchronization

The reason of the aforesaid limitations is that the current criteria cannot reflect the specific process of the communication. Shared resources, encoding, and synchronization are not included in the current threat estimation criteria, which are not enough to measure threats of covert channels. Therefore, the anti-detection criterion is utilized to estimate the covertness of covert channels and to be a supplement for the current threat estimation criteria. The anti-detection criterion is detailed in Section 5. Based on this criterion, the paper presents a novel covert channel restricting algorithm which aims at obtaining the right balance between the system performance and the security enforcement.

5 Anti-Detection Criterion

This section provides formal definitions of the anti-detection criterion and analyzes each element of the criterion. Functions for evaluating anti-detection capacity of practical nature are also discussed.

5.1 Anti-Detection Criterion Definition

To analyze the key elements that have influence on the anti-detection capacity of covert channels, various components of covert channels are required to be stated comprehensively for understanding the contributions of this work. Based on the previous work[2,19], the formal definition of covert channel can be concluded.

Definition 1 (Covert Channel). *A covert channel can be expressed as the following triple.*

$$CC = (SR, PA_h, PV_i), \quad (1)$$

where SR represents shared resources (e.g., shared variables and caches) in computer systems, PA_h is an entity at the high security level that modifies SR, and PV_i is a low security level entity that observes changes of SR. The communication from PA_h to PV_i is prohibited by the system's security policy. In Linux, PA_h and PV_i correspond to two processes of respective users at different security levels.

Definition 1 gives a clear and integral description of transmitting participants of covert channels, but it lacks significant details such as operations on shared resources that are needed for further analysis of characteristics of covert channels. To understand the behavior of covert channels, we start our study with specific communication process and expand the above-mentioned three elements separately.

Previous covert channels are limited to one single shared resource; however, with continuous evolution of covert channels, attackers can take advantages of multiple shared resources simultaneously in one information communication process[9,20] such as the covert communication protocols discussed in Subsection 4.2. Therefore, when analyzing the elements of shared resources, all the resources involved in covert information transmission need to be considered comprehensively. The shared resource, SR, can be further extended as follows.

Definition 2 (Shared Resource). *SR is a collection of:*

$$SR = (R_1, R_2, ..., R_n). \quad (2)$$

As highlighted in Definition 1, the essential functions of entities PA_h and PV_i are to read and write shared resources, respectively. However, the required method to realize these functions usually corresponds to one operation, for example, the system call $fork()$ responding to both the write and the read operations on *last_pid* in Linux, except that the number of calls in the read process and the write process is different. However, to analyze the anti-detection capability of covert channels, the summary and statistics of its behaviors are needed, rather than distinctive objectives of respective operations. Hence, our work does not distinguish between a read and a write operation at all.

From the aforementioned discussion, another element of the anti-detection criterion, *Operation*, is deduced as follows.

Definition 3 (Operation). *Operation denotes the read and the write operations on shared resources*:

$$Operation = (O_1, O_2, ..., O_n). \quad (3)$$

To ensure a successful covert transmission, the sender, the receiver, the synchronization mechanism, and the encoding mechanism need to perform the specified operation(s) on the target shared resource(s). For example, in order to send one bit, the sender may perform an operation O_i (e.g., write) on a shared resource R_i such as *last_pid*. Thus, the entire *Operation* must be divided into several parts according to the type of shared resources, and the corresponding mapping relationship between *Operation* and SR needs to be set up accordingly. Summarizing (2) and (3), *Operation* is mapped to SR as follows:

$$CCMap = (R_1(O_1, ..., O_n), ..., R_n(O_1, ..., O_n)). \quad (4)$$

Different combinations (mappings) of the elements in (4) generate diverse covert channels which differ in synchronization mechanisms, encodings, and sender/receiver behaviors. Compared with (1), (4) is more specific systematic in describing the relationship between shared resources and operations. (4) also covers the resources needed for structuring any sort of covert channels, which can be used by attackers to make endless attempts to combine and develop the resources in order to derive distinctive attack strategies in spatial and time dimension.

It can be seen as a limitation that (4) is confined to the static division and depiction of significant elements of covert channels. Moreover, (4) only models the behaviors of the sender, the receiver, and the key factors in the communication process, and it cannot truly quantify the threats of the covert channel. In order to acquire a concrete result of the threat estimation, the elements in $CCMap$ need to be quantified by relevant calculative strategies. Therefore, the anti-detection criterion (ADC) is defined as follows:

$$ADC = (\Theta(CCMap)), \tag{5}$$

where Θ is a calculation method to quantify the threat and covertness of a covert channel and will be detailed in Subsection 5.3.

5.2 Analysis of ADC

In this subsection, we discuss ADC under three possible scenarios.

1) *Complexity and Covertness*. On the one side, when a communication protocol that the attackers adopt in a channel becomes more complex, it also behaves more remarkably in terms of accuracy or capacity. However, this scenario also means a high risk in the anti-detection criterion on the other side[9,36,53], because a complex protocol may perform a nontrivial set of operations on shared resources, which would indicate that the attacker is active in a system. Thus, the protocol is more easily detectable under this circumstance. The most extreme situation would be the case when an attacker does not challenge the system and hence the risk of being detected is theoretically zero at this point. In such cases, ADC score approaches infinity. For instance, the two covert channels TCTP and SAP discussed in [9, 36] can be a concrete example to illustrate our perspectives. SAP uses more operations on the same shared resources and is also more complicated than TCTP. Under the surrounding equipped with the covert channel restriction strategies, the accuracy of SAP is 97% while the accuracy of TCTP is merely 51%[9,36]. Therefore, covert information transmitted by SAP can be correctly decoded while that from TCTP is useless. However, as compared with TCTP, SAP is easier to be detected by monitoring the respective usage of corresponding shared resources.

It is worth mentioning that the complexity of a covert channel depends on the number of shared resources, the usage (frequency of use) of shared resources, the synchronization mechanism, and the encoding mechanism. Thus, more shared resources result in highly complex covert channels due to the number of additional system/function calls. In other words, as the number of shared resources increases, the complexity of the protocol becomes higher. Any covert channel can be detected as long as the abnormal behavior of any function call of the channel is detected.

Furthermore, synchronization mechanisms also improve the accuracy of a covert transmission as well as the complexity of a policy. This is due to the fact that extra shared resources or the increased usage of currently existing shared resources is needed to synchronize. Impacts of encoding on protocols depend on how it affects shared resources. As a concluding remark, the number and the usage of shared resources are the key factors that affect the ADC, complexity, and covertness of a channel. Meanwhile, the complexity is a key element to quantify ADC and will be defined in Subsection 5.3.

2) *Distinguishably*. The anti-detection criterion should be sensitive enough to distinguish different covert channels[53]. That is, the difference between ADC scores of different communication protocols should be significantly large. In case two different protocols get the same ADC, then the proposed criterion cannot provide meaningful guidance for threat estimation. On the contrary, when the ADC of two similar protocols is significantly different, it indicates that the difference between the two protocols is huge. Eventually, the defender can then choose to eliminate the most threatening covert channels while aiming at reducing the performance impact on the overall system functionality.

3) *Key Elements of the Communication Process*. The anti-detection criterion should reflect the process of channel communications. As already indicated, the process of communications is not considered by the current threat estimation criteria. There exist cases where the threat and the difference between different protocols cannot be estimated (see Subsection 4.3). Therefore, shared resources, encoding, and synchronization are critical factors to be included in an anti-detection criterion.

5.3 Calculation Policy

According to (5), the calculation policy Θ should meet the following aspects: 1) quantifying every *Operation* on a single shared resource (or multiple shared resources) in (2) and the mappings between *Operation* and SR, and 2) giving a specific threat estimation score. Specifically, Θ must include two factors mentioned in Subsection 5.2: the number of shared resources, and the usage of shared resources such as the number and regularity of the operations on shared resources. Thus, two methods are provided below to compute Θ.

Θ: RP (*Restriction Proportion*). RP quantifies the usage of each shared resource from the perspective of the number of the operations on the shared resource.

RP includes the extra usage of shared resources brought by covert channels. Meanwhile, the higher the extra usage of the shared resource, the more important the shared resource in the covert channel. Let m be the number of all operations on a shared resource R_i in the system, and n be the number of operations on R_i in the covert channel. In order to consider the impact of the operations introduced by the covert channel on the overall operation of the system, we should concentrate on the ratio of n to m rather than n itself. For a shared resource R_i, RP_i is given by:

$$RP_i = n/m. \tag{6}$$

Θ: TIV (*Time Interval Variance*). TIV quantifies the usage of each shared resource from the perspective of the regularity of the operations on the shared resource. Considering the regularity of occurrences of *Operation* on shared resources, TIV is defined as the variance of time intervals between operations. Let t_j $(0 < j \leqslant n)$ denote the time interval between two consecutive operations on a shared resource R_i in the covert channel. Set $S = \{t_1, t_2, ..., t_n\}$ consists of all the time intervals in the covert channel and M is the average of all values in S. For a shared resource R_i, TIV_i is depicted by:

$$TIV_i = ((t_1 - M)^2 + (t_2 - M)^2 + ... + (t_n - M)^2)/n. \tag{7}$$

Therefore, the corresponding formal definition of ADC in (5) is further extended as follows:

$$\begin{aligned} & ADC \\ = & (RP(R_1(O_1, ..., O_n)) \wedge TIV(R_1(O_1, ..., O_n)) \wedge ... \wedge \\ & RP(R_n(O_1, ..., O_n)) \wedge TIV(R_n(O_1, ..., O_n))), \end{aligned} \tag{8}$$

where RP and TIV denote the way that Θ quantifies all the combinations of *Operation* and SR. $\wedge$ is a union to integrate the intermediate results. The way to compute $\wedge$ is detailed in (10).

Complexity. According to the aforementioned discussions in Subsection 5.2, the complexity of a covert channel contains both the number and the usage of shared resources involved in a covert channel in order to quantify the covertness and ADC of a covert channel. Therefore, the complexity should integrate all the RP and TIV results. Specifically, let $VarNum$ equal n in (2). In order to know whether the covert channel is active from the perspective of the entire system, it is more appealing to consider the ratio of n to m as introduced in (6) rather than n itself. Therefore, RP has been adopted to quantify the times the shared resources are used. Similarly, to illustrate the gap between variances, the weight factor[54] is adopted to estimate the time characteristics of a shared resource by the following formula:

$$WeightTIV_i = \frac{TIV_{\min}}{TIV_i} (0 < i \leqslant k),$$

where $TIV_{\min}$ is the minimum TIV of a shared resource in all protocols, and k is the number of the protocols. The synchronization mechanism and the encoding mechanism can be represented by the *Operation* on SR and therefore are included in $VarNum$ and RP. Thus, the complexity is defined as follows:

$$Complexity = \sum_{i=1}^{VarNum} (RP_i + WeightTIV_i). \tag{9}$$

According to the three standards proposed in Subsection 5.2, the complexity of a covert channel is inversely proportional to its anti-detection ability. Therefore, ADC should be inversely proportional to the complexity. Finally, ADC is given by:

$$ADC = 1/ \sum_{i=1}^{VarNum} (RP_i + WeightTIV_i). \tag{10}$$

6 Covert Channel Restriction

A majority of related work in the covert channel[2,18,19] assumes that covert channels cannot be completely eliminated. The standard such as CC and TCSEC also allows using compromised restriction strategies as trade-offs between the security and the performance of the system. Therefore, the assumptions to explain our work more precisely are shown as follows. 1) More than one covert channel exists in the system. In this paper, the three real covert channels (to be detailed in Subsection 6.1) at different threat levels have been adopted. 2) Channel restriction strategies have been designed for each channel. This is a reasonable assumption since adding noises[12,41] to a channel is a common approach and easy to implement. 3) Deploying all the restriction strategies will incur high performance overheads such that it can seriously affect the system usability[2,18,19]. Considering the system usability, the covert channel with the biggest threat should be restricted first compared with the other channels. The remaining channels with less threats but not covered by the restriction can be solved with other solutions such as covert channel audit.

The goals of VMCSCR are described as follows: 1) ranking the threat of the intra-VM covert storage channels in the system; 2) mitigating the threat that the channels bring about while maintaining the usability and resource efficiency of computer systems based on the ranking. Additionally, ADC is used to quantify the threat of covert channels. The effectiveness of VMCSCR was evaluated by comparing the technique with classic restriction policy Pump[29,43].

6.1 Three Real Covert Channel Scenarios

Three real communication protocols are utilized to illustrate the limitations of current threat estimation criteria and the effectiveness of the anti-detection criterion. Two programs named *CCsender*() and *CCreceiver*() are implemented as the sender and the receiver, respectively. The three protocols are detailed as follows.

Protocol 1 is detailed in Subsection 4.2. To be specific, two system calls $fork()$ and *create*() are mapped to the operations on shared resources *last_pid* and temporary files, respectively.

Protocol 2 is also described in Subsection 4.2. The time T is set to 500 µs.

The only difference between the two communication protocols is the synchronization mechanism. Without loss of generality, this paper uses a classic communication protocol without the synchronization for comparison. That is, there is no feedback from a receiver to a sender when receiving each bit during the transmission. A sender does not know whether the receiver has got the information. There may be overlap between the transmissions of two continuous bits.

Protocol 3 is detailed in Algorithm 1.

Algorithm 1. Protocol 3

```
   Data: BitsSend: all the bits to be transmitted
   Result: BitsRecveive: all the received bits
1. Sender: add 3 to last_pid by calling $fork()$ three times;
   Receiver: observe the change and start to receive bits;
2. Sender: if the bit to be sent in BitsSend is binary 0, then
   do nothing. If it is 1, then add 2 to last_pid;
   Receiver: observing the change of last_pid and recording
   the bit to BitsRecveive.;
   Repeat this step until all the bits have been transmitted;
3. Sender: add 3 to last_pid by calling $fork()$ three times;
   Receiver: observe the change and stop receiving bits;
4. End the transmission.
```

6.2 ADC of the Three Protocols

Channel capacity is an important metric for estimating threat and transmission ability of covert channels. Therefore the channel capacity of the three different protocols is measured 20 times and the average is taken. Our experimental evaluation results in Fig.1 show that the channel capacity of the three protocols is nearly 2 Kbps. The channel capacity of protocol 3 is slightly lower than that of the other two protocols. However, the order of the channel capacity of the three protocols is not fixed. It is worth mentioning that the channel capacity of protocol 2 can be intentionally adjusted by simply changing time T. Therefore, it is hard to decide which protocol is more threatening. The anti-detection criterion can eliminate this limitation.

The parameters in RP and TIV are obtained by the Linux Audit②. Linux Audit is a toolkit maintained by the Linux Open Source Community for recording key system calls and operations. It can accurately record critical information such as the number of operations on shared resources and the time the operations occurred. For each channel, the RP_i and TIV_i values for each of its shared variables are calculated based on (6) and (7) respectively. Next, (9) and (10) are utilized to compute the complexity and ADC respectively. RP_i, RP_{sum}, TIV_i, $WeightTIV_{\text{sum}}$ of the three protocols are listed in Table 2. $WeightTIV_{\text{sum}}$ is the sum of all $WeightTIV_i$, and RP_{sum} is the sum of all RP_i. According to (9), the complexity of the three protocols is sorted as follows: protocol 1 (2.52) > protocol 2 (2.31) > protocol 3 (1.17). The sort of the ADC scores is: protocol 1 (0.39) < protocol 2 (0.43) < protocol 3 (0.86). Although the three covert channels have nearly the same capacity, the ADC scores show that protocols 1 and 2 are much easier to detect than protocol 3.

6.3 Covert Channel Restriction Algorithm

The goal of covert channel restriction is to reduce the threats of channels and keep it at a low level permitted by the security criteria[2,19]. Since the covert information transmission is based on the legitimate operations, eliminating all covert channels may lead to system crash. Keeping this point in mind, the priority should be given to the restriction for a covert channel that posses the threat. Based on the above discussions, VMCSCR uses the anti-detection criterion to mitigate threats of covert channels while maintaining the resource efficiency of the system. VMCSCR applies optimizations techniques to guarantee both the security and the resource efficiency for the system.

As stated previously, a more complex channel can achieve higher capacity and accuracy but with low ADC

②Linux Audit. http://people.redhat.com/sgrubb/audit/, Sept. 2019.

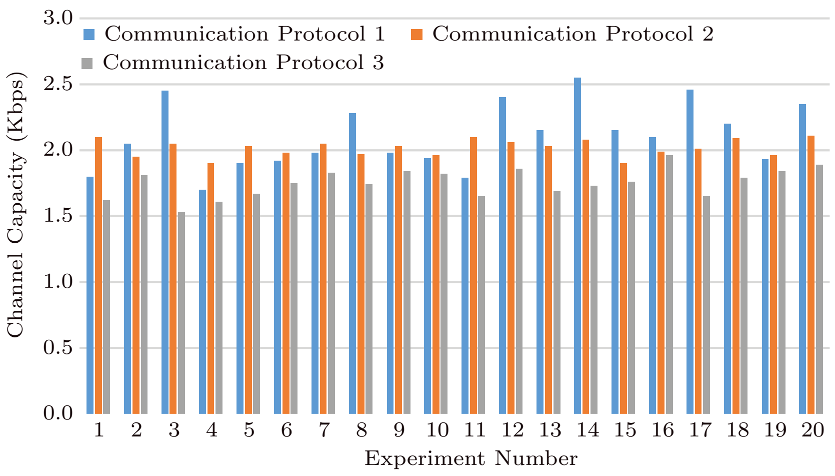

Fig.1. Channel capacity of protocols 1, 2, and 3.

score. Such a channel consumes lots of system resources and incurs high performance overheads. Meanwhile, if there are several covert channels with different levels of threat, more resources should be allocated for the restriction strategies utilized to restrict the most threatening channels due to the system performance limitations. Therefore, an appropriate trade-off between the system security and the system performance can be obtained while deploying restriction strategies for each covert channel by using ADC.

Table 2. RP, TIV, and $WeightTIV$ of the Three Communication Protocols

Protocol	RP_1	RP_2	RP_{sum}	TIV_1	TIV_2	$WeightTIV_{\text{sum}}$
1	0.26	0.49	0.75	3.23	3.78	1.77
2	0.33	0.13	0.46	3.82	2.91	1.85
3	0.35	-	0.35	3.96	-	0.82

Furthermore, each channel may use multiple shared resources for covert information transmission. In order to further mitigate the system performance overhead, it is necessary to restrict the shared resources that perform the most operations. The RP of ADC represents the weight of *Operation* on each shared resource and also shows the impacts of *Operation* on system performance. Therefore, restricting some of the most important shared resources based on RP can reduce performance overhead instead of the restriction on all shared resources. There can be situations where only a single covert channel exists in the system. This is just a special case of multiple covert channels. VMCSCR can optimize its shared resources restriction based on the RP value mentioned above.

We now detail our channel restriction algorithm VMCSCR in Algorithm 2, where $CC[n]$ is a set of n covert channels in the system and $RS[n]$ is a set of pre-designed restriction strategies for each channel accordingly. The VMCSCR algorithm can be briefly described as follows. 1) For all the covert channels to be processed, calculate their RP and TIV. 2) For each covert channel, calculate its ADC. 3) For all the covert channels, rank the covert channels $CC[n]$ with their ADC scores in ascending order. 4) For the channel $CC[i]$, deploy $RS[i]$ by introducing noises through periodic random trigger operations on the unprocessed shared resource $SR[k]$ which has the maximum RP value. $RS[i]$ should reduce the accuracy of $CC[i]$ to a certain threshold *threshold* such as 50%. 5) The extra operations (performance overheads) brought by $RS[i]$ are recorded by Linux Audit and are denoted as OP_i for $CC[i]$. 6) For the channel $CC[i]$, add OP_i to OP_{sum} to get the sum of operations introduced by the current deployed restriction strategies. If OP_{sum} is less than the threshold value, go back to step 4. Otherwise, stop the algorithm and return the results.

Algorithm 2. VMCSCR Algorithm

```
Data: $CC[n]$: covert channels; $RS[n]$: restriction
      strategies; $SR[n]$: shared resources
Result: the restricted $CC[n]$
while $CC[++i]$ is not empty do
    Calculate $TIV[i]$ for $CC[i]$;
    Calculate $RP[i]$ for $CC[i]$;
    Calculate $ADC[i]$ for $CC[i]$ with $TIV[i]$ and $RP[i]$;
end
Sort $CC[n]$ based on the ADC scores;
while $CC[++j]$ is not empty &&($OP_{\text{sum}} \leqslant threshold$)
do
    $SR[j]_k = GetMaxSR(SR[j], RP[j])$;
    $ExE(RS[j], CC[j], SR[k])$;
    $OP_j = CaculateWithLinuxAudit(RS[j], CC[j])$;
    $OP_{\text{sum}} = OP_{\text{sum}} + OP_j$;
end
```

6.4 Experimental Evaluations

Evaluations were carried out on a VM with 1.9 GHz CPU, 16 GB memory, and 1 TB internal storage. The testbed system in the VM is a Linux operating system (Linux x86_64) based on Linux kernel 4.2.0. The VM was deployed on our campus server, which was not isolated from our experiments, i.e., other users were utilizing it at the same time. Thus, the environment provided is a realistic scenario and mimics a public cloud. Although there were noises introduced from shared components by other co-located VMs, we could not guarantee that the noise exists all the time (e.g., midnights or weekends). Without loss of generality, the system ran a program that randomly modified shared resources while the time interval for modifications was randomly selected from 1 ms to 2 ms. The datasets adopted in this paper are three realistic covert storage channels detailed in Subsection 6.1. Our work was to restrict the threat of the three channels.

The VMCSCR algorithm was evaluated with both the accuracy and the performance of covert channel transmissions. When a restriction policy made the accuracy of transmission results significantly degraded, the information transmitted by the channel could not be decoded correctly by a receiver. Editing distance was used to estimate the accuracy of covert channel as reported in [24]. Then, in order to evaluate the performance impacts on systems brought by VMCSCR, Linux Audit was used to monitor the usage of shared resources.

6.4.1 *Effectiveness of VMCSCR in Threat Restriction*

The three covert channels mentioned in Subsection 6.1 were the input $CC[n]$ of Algorithm 2. $SR[n]$ contains *last_pid* and temporary files. Restriction strategies $RS[n]$ interfere with covert transmissions by adding noises to covert channels. Specifically, it is worth mentioning that $RS[n]$ is a set of well-designed programs that randomly trigger operations on shared resources every 300 µs–700 µs, and therefore change the value of *last_pid* or the status of temporary files.

Accuracy is adopted to evaluate the effectiveness of VMCSCR. As Cabuk *et al.*[24] pointed out, the accuracy is computed by the edit distance[55], which represents the minimum distance between two strings. Fig.2 reveals the accuracy of the three covert communication protocols with and without VMCSCR, respectively. The upper three lines in Fig.2 represent the accuracy of the three channels in a normal environment, and nearly all the accuracy reaches 80%. The lower threes lines in Fig.2 represent the accuracy of the three channels with VMCSCR, and nearly all the accuracy rate is reduced to 60% or less. The average accuracy of the three protocols is depicted in Table 3. VMCSCR significantly reduces the accuracy, and the highest accuracy is merely 52.43%. In such an accuracy rate, the information decoded by the receiver is useless, which

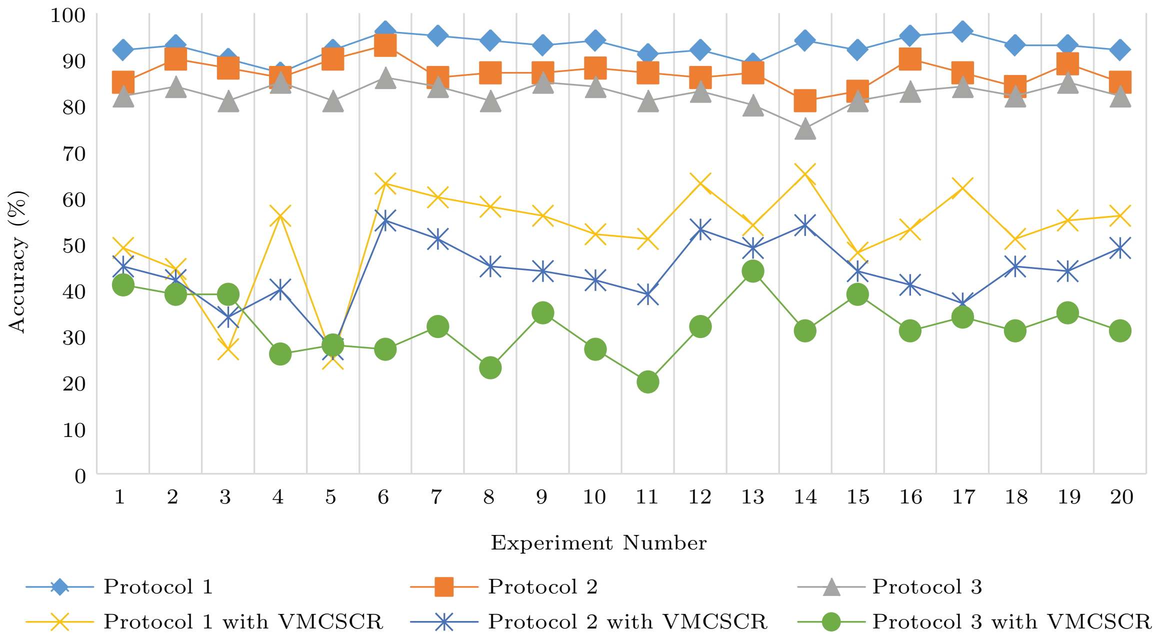

Fig.2. Accuracy of the three communication protocols in normal/restricted environment.

shows the effectiveness of VMCSCR and hence the intended purpose of our work.

Table 3. Accuracy of the Three Communication Protocols in Normal/Restricted Environment on Average

Protocol	Accuracy (%)
Protocol 1	92.65
Protocol 2	86.95
Protocol 3	82.45
Protocol 1 with VMCSCR	52.43
Protocol 2 with VMCSCR	44.00
Protocol 3 with VMCSCR	32.25

6.4.2 *Effectiveness of VMCSCR in Performance Overhead Mitigation*

Restriction strategies reduce threats of covert channels while imposing extra overhead on system performance. The number of operations on shared resources required by the restriction policy is used to measure the performance overhead. It is observed that the system overhead lowers with the reduced number of operations required by a policy. To evaluate the performance of our work in a practical scenario, we compare VMCSCR with the classical restriction policy Pump[29,43]. It is worth noticing that Pump is adopted by the US Naval Research Laboratory[2] and deployed in practice. The system overhead and optimization are shown by comparing the numbers of operations used by VMCSCR and Pump.

The settings of VMCSCR and Pump were well-designed to ensure that both strategies introduce the same level of noise into the channel. Specifically, for each covert channel, the parameters of VMCSCR and Pump were adjusted such that the accuracy of the covert channel was reduced to 50% when the corresponding restriction policy was deployed. Then, the system performance overheads brought by the VMCSCR or Pump were recorded by Linux Audit for the comparison. As shown in Fig.3, the horizontal axis represents seven covert channel scenarios based on combinations of the three protocols in Subsection 6.1. The vertical axis shows the percentage of operations reduced by VMCSCR compared with Pump. According to Linux Audit, VMCSCR consumes less operations on the shared resources than Pump. When only the protocol 2 exists ((2) in Fig.3), VMCSCR reduces 71.14% of the operations for introducing noises into the channel compared with Pump, and they both limit the accuracy of the channel to 50%. VMCSCR did not introduce any optimizations to system performance when only protocol 3 exists. This is due to the fact that VMCSCR is an optimized restriction policy for multiple channels or a single channel with multiple shared resources. It should be noted that protocol 3 only contains one shared resource and hence the effectiveness of VMCSCR is similar to that of Pump in this case. Consequently, when covert channels contain multiple shared resources, VMCSCR significantly reduces the performance overhead which is aligned with our theoretical results established in this work.

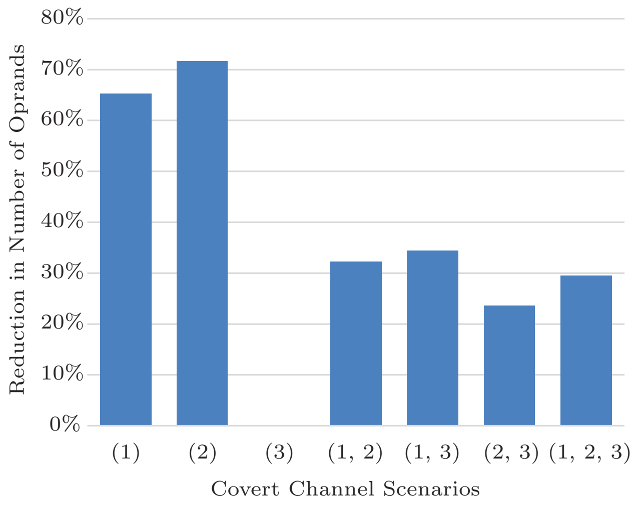

Fig.3. Experimental results of VMCSCR.

7 Conclusions

With the development of virtualization techniques in cloud computing paradigm, covert storage channels are the threats arising from the adoption of information hiding techniques in VMs. Therefore, covert channel threat estimation and restriction are critical to defend such attacks. In this paper, two real covert communication scenarios from previous research were used to demonstrate the limitations of the current threat estimation criteria. A novel anti-detection criterion was proposed to eliminate these limitations, and it was treated as a supplement for the current threat estimation criteria. The anti-detection criterion was formally defined, analyzed, and evaluated through three real communication protocols. A covert channel restriction algorithm VMCSCR based on anti-detection criterion was presented to mitigate intra-VM covert storage channel attacks. It was noticed that VMCSCR caused sufficiently high error rates to covert channels and achieved a proper trade-off between the threat restriction and system performance overheads. Experiments demonstrated the effectiveness of VMCSCR by comparing VMCSCR with Pump in three real covert channel scenarios. As future work, the possibility of applying VMCSCR in covert timing channels can be explored. Our future work also aims at performing covert channel analysis in mobile operating systems and other emerging areas[2] with anti-detection criterion and VMCSCR.

References

[1] Lampson B W. A note on the confinement problem. *Commun. ACM*, 1973, 16(10): 613-615.

[2] Wang Y, Wu J, Zeng H, Ding L, Liao X. Covert channel research. *Journal of Software*, 2010, 9(21): 2262-2288. (in Chinese)

[3] Yan M, Shalabi Y, Torrellas J. ReplayConfusion: Detecting cache-based covert channel attacks using record and replay. In *Proc. the 49th Annual IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2016, Article No. 39.

[4] Archibald R, Ghosal D. Design and analysis of a model-based covert timing channel for skype traffic. In *Proc. IEEE Conf. Communications and Network Security*, Sept. 2015, pp.236-244.

[5] Chard K, Caton S, Rana O, Bubendorfer K. Social cloud: Cloud computing in social networks. In *Proc. the 3rd IEEE Int. Conf. Cloud Computing*, July 2010, pp.99-106.

[6] Riaz A, Qadir J, Younis U, Rasool U R, Ahmad H F, Kiani A K. Intrusion detection systems in cloud computing: A contemporary review of techniques and solutions. *Journal of Information Science and Engineering*, 2017, 33(3): 611-634.

[7] Wang Z, Hayat M M, Ghani N, Shaban K B. A probabilistic multi-tenant model for virtual machine mapping in cloud systems. In *Proc. the 3rd IEEE Int. Conf. Cloud Networking*, Oct. 2014, pp.339-343.

[8] Win T Y, Tianfield H, Mair Q, Said T A, Rana O F. Virtual machine introspection. In *Proc. the 7th Int. Conf. Security of Information and Networks*, September 2014, Article No. 405.

[9] Lin Y, Malik S U R, Bilal K, Yang Q, Wang Y, Khan S U. Designing and modeling of covert channels in operating systems. *IEEE Transactions on Computers*, 2016, 65(6): 1706-1719.

[10] Kadloor S, Kiyavash N, Venkitasubramaniam P. Mitigating timing side channel in shared schedulers. *IEEE/ACM Trans. Netw.*, 2016, 24(3): 1562-1573.

[11] Evtyushkin D, Ponomarev D, Abu-Ghazaleh N. Understanding and mitigating covert channels through branch predictors. *ACM Trans. Archit. Code Optim.*, 2016, 13(1): Article No. 10.

[12] Zhang R, Su X, Wang J, Wang C, Liu W, Lau R W H. On mitigating the risk of cross-VM covert channels in a public cloud. *IEEE Transactions on Parallel and Distributed Systems*, 2015, 26(8): 2327-2339.

[13] Ristenpart T, Tromer E, Shacham H, Savage S. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. In *Proc. the 16th ACM Conf. Computer and Communications Security*, November 2009, pp.199-212.

[14] Wu Z, Xu Z, Wang H. Whispers in the hyper-space: High-bandwidth and reliable covert channel attacks inside the cloud. *IEEE/ACM Transactions on Networking*, 2015, 23(2): 603-615.

[15] Betz J, Westhoff D. C^3-sched — A cache covert channel robust cloud computing scheduler. In *Proc. the 9th Int. Conf. Internet Technology and Secured Transactions*, Dec. 2014, pp.54-60.

[16] Oren Y, Kemerlis V P, Sethumadhavan S, Keromytis A D. The spy in the sandbox: Practical cache attacks in JavaScript and their implications. In *Proc. the 22nd ACM SIGSAC Conf. Computer and Communications Security*, Oct. 2015, pp.1406-1418.

[17] Zhang X, Xiao Y, Zhang Y. Return-oriented flush-reload side channels on ARM and their implications for Android devices. In *Proc. the 2016 ACM SIGSAC Conf. Computer and Communications Security*, Oct. 2016, pp.858-870.

[18] Moskowitz I S, Kang M H. Covert channels-here to stay? In *Proc. the 9th IEEE Annual Conf. Computer Assurance*, Jun 1994, pp.235-243.

[19] Zander S, Armitage G, Branch P. A survey of covert channels and countermeasures in computer network protocols. *IEEE Communications Surveys Tutorials*, 2007, 9(3): 44-57.

[20] Wendzel S, Zander S, Fechner B, Herdin C. Pattern-based survey and categorization of network covert channel techniques. *ACM Comput. Surv.*, 2015, 47(3): Article No. 50.

[21] Johnson D, Lutz P, Yuan B. Behavior-based covert channel in cyberspace. In *Proc. the 4th International ISKE Conference on Intelligent Systems and Knowledge Engineering*, Nov. 2009, pp.311-318.

[22] Wu J, Wang Y, Ding L, Liao X. Improving performance of network covert timing channel through Huffman coding. *Mathematical and Computer Modelling*, 2012, 55(1): 69-79.

[23] Ksentini A, Abassi O. A comparison of VoIp performance over three routing protocols for IEEE 802.11s-based wireless mesh networks (wlan mesh). In *Proc. the 6th ACM Int. Symp. Mobility Management and Wireless Access*, Oct. 2008, pp.147-150.

[24] Cabuk S, Brodley C E, Shields C. IP covert timing channels: Design and detection. In *Proc. the 11th ACM Conf. Computer and Communications Security*, Oct. 2004, pp.178-187.

[25] Maqbool Q, Ayub S, Zulfiqar J, Shafi A. Virtual TCAM for data center switches. In *Proc. IEEE Conf. Network Function Virtualization and Software Defined Network*, Nov. 2015, pp.61-66.

[26] Tahir R, Khan M T, Gong X, Ahmed A, Ghassami A, Kazmi H, Caesar M, Zaffar F, Kiyavash N. Sneak-peek: High speed covert channels in data center networks. In *Proc. the 35th Annual IEEE Int. Conf. Computer Communications*, April 2016, Article No. 138.

[27] Wang M, Wu Q, Qin B, Wang Q, Liu J, Guan Z. Lightweight and manageable digital evidence preservation system on bitcoin. *Journal of Computer Science and Technology*, 2018, 33(3): 568-586.

[28] Zou M H, Ma K, Wu K J, Sha E H M. Scan-based attack on stream ciphers: A case study on eSTREAM finalists. *Journal of Computer Science and Technology*, 2014, 29(4): 646-655.

[29] Wu J, Ding L, Lin Y, Min-Allah N, Wang Y. XenPump: A new method to mitigate timing channel in cloud computing. In *Proc. the 5th IEEE Int. Conf. Cloud Computing*, June 2012, pp.678-685.

[30] Goguen J A, Meseguer J. Unwinding and inference control. In *Proc. the 1984 IEEE Symp. Security and Privacy*, April 1984, pp.75-87.

[31] Denning D E. A lattice model of secure information flow. *Commun. ACM*, 1976, 19(5): 236-243.

[32] Kemmerer R A. A practical approach to identifying storage and timing channels: Twenty years later. In *Proc. the 18th Annual Computer Security Applications Conf.*, Dec. 2002, pp.109-118.

[33] Wu J, Ding L, Wang Y, Han W. A practical covert channel identification approach in source code based on directed information flow graph. In *Proc. the 5th Int. Conf. Secure Software Integration and Reliability Improvement*, June 2011, pp.98-107.

[34] Millen J. 20 years of covert channel modeling and analysis. In *Proc. the 1999 IEEE Symp. Security and Privacy*, May 1999, pp.113-114.

[35] Wu J, Wang Y, Ding L, Zhang Y. Constructing scenario of event-flag covert channel in secure operating system. In *Proc. the 2nd Int. Conf. Information and Multimedia Technology*, Dec. 2010, pp.371-375.

[36] Lin Y, Ding L, Wu J, Xie Y, Wang Y. Robust and efficient covert channel communications in operating systems: Design, implementation and evaluation. In *Proc. the 7th IEEE Int. Conf. Software Security and Reliability*, June 2013, pp.45-52.

[37] Zeng H, Wang Y, Zu W, Cai J, Ruan L. New definition of small message criterion and its application in transaction covert channel mitigating. *Journal of Software*, 2009, 20(4): 985-996. (in Chinese)

[38] Cabuk S, Brodley C E, Shields C. IP covert channel detection. *ACM Trans. Inf. Syst. Secur.*, 2009, 12(4): Article No. 22.

[39] Wang C, Zhang C, Wu B, Tan Y, Wang Y. A novel anti-detection criterion for covert storage channel threat estimation. *Science China Information Sciences*, 2018, 61(4): Article No. 048101.

[40] Tsai C R, Gligor V D. A bandwidth computation model for covert storage channels and its applications. In *Proc. the 1988 IEEE Symp. Security and Privacy*, Apr. 1988, pp.108-121.

[41] Wu J, Ding L, Wu Y, Min-Allah N, Khan S U, Wang Y. C^{2}Detector: A covert channel detection framework in cloud computing. *Sec. and Commun. Netw.*, 2014, 7(3): 544-557.

[42] Ristad E S, Yianilos P N. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998, 20(5): 522-532.

[43] Kang M H, Moskowitz I S. A pump for rapid, reliable, secure communication. In *Proc. the 1st ACM Conf. Computer and Communications Security*, Nov. 1993, pp.119-129.

[44] Zhai J T, Wang M Q, Liu G J, Dai Y W. Detecting *jitterbug* covert timing channel with sparse embedding. *Security and Communication Networks*, 2016, 9(11): 1509-1519.

[45] Gianvecchio S, Wang H. Detecting covert timing channels: An entropy-based approach. In *Proc. the 2007 ACM Conf. Computer and Communications Security*, Oct. 2007, pp.307-316.

[46] Hunger C, Kazdagli M, Rawat A, Dimakis A, Vishwanath S, Tiwari M. Understanding contention-based channels and using them for defense. In *Proc. the 21st IEEE Int. Symp. High Performance Computer Architecture*, Feb. 2015, pp.639-650.

[47] Zhang R, Su X, Wang J, Wang C, Liu W, Lau R W H. On mitigating the risk of cross-VM covert channels in a public cloud. *IEEE Transactions on Parallel and Distributed Systems*, 2015, 26(8): 2327-2339.

[48] Caviglione L, Podolski M, Mazurczyk W, Ianigro M. Covert channels in personal cloud storage services: The case of dropbox. *IEEE Transactions on Industrial Informatics*, 2017, 13(4): 1921-1931.

[49] Gai K, Wu Y, Zhu L, Xu L, Zhang Y. Permissioned blockchain and edge computing empowered privacy-preserving smart grid networks. *IEEE Internet of Things Journal*. doi:10.1109/JIOT.2019.2904303.

[50] Evtyushkin D, Ponomarev D, Abu-Ghazaleh N. Understanding and mitigating covert channels through branch predictors. *ACM Trans. Archit. Code Optim.*, 2016, 13(1): Article No. 10.

[51] Kim T, Peinado M, Mainar-Ruiz G. STEALTHMEM: System-level protection against cache-based side channel attacks in the cloud. In *Proc. the 21st USENIX Security Symp.*, Aug. 2012, pp.189-204.

[52] Zhang Y, Juels A, Oprea A, Reiter M K. HomeAlone: Co-residency detection in the cloud via side-channel analysis. In *Proc. IEEE Symp. Security and Privacy*, May 2011, pp.313-328.

[53] Lin Y. Research on the covert channel analysis of general and cross platform technology [Ph.D. Thesis]. Institute of Software, Chinese Academy of Sciences, 2016. (in Chinese)

[54] Xu C J, Ding K H, Cai J Q, Grafarend E W. Methods of determining weight scaling factors for geodetic-geophysical joint inversion. *Journal of Geodynamics*, 2009, 47(1): 39-46.

[55] Ristad E S, Yianilos P N. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998, 20(5): 522-532.

Chong Wang is currently working toward his Ph.D. degree at the Institute of Software, Chinese Academy of Sciences, Beijing. He received his B.S. degree in computer science and technology from Beijing University of Technology, Beijing, in 2014. His main research interests include covert channel analysis, virtualization techniques, and information hiding.

Nasro Min-Allah is an associate professor at the College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, Dammam, since September 2014. He received his Ph.D. degree in real-time systems from the Graduate University of the Chinese Academy of Sciences, Beijing, in 2008. He worked at the SuperTech Group of the MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL) from September 2012 to June 2014 as a visiting scientist and taught at the Electrical Engineering and Computer Science Department of Massachusetts Institute of Technology (EECS-MIT) from January 2013 to May 2014.

Bei Guan received his B.S. degree in computer science from Tianjin University, Tianjin, in 2007, and his Ph.D. degree in computer software and theory from the Institute of Software, Chinese Academy of Sciences, Beijing, in 2014. His primary research interests include operating system techniques, virtualization techniques, cloud computing, and system security. He focuses on the management and scheduling of virtual resource in cloud computing now.

Yu-Qi Lin received his Ph.D. degree in computer software and theory from the Institute of Software, Chinese Academy of Sciences, Beijing, in 2016. He is currently working at Block Chain Research Center of Blue Helix, Shanghai. His primary research interests include covert channel analysis, virtualization techniques, cloud computing, and system security.

Jing-Zheng Wu received his Ph.D. degree in computer software and theory from the Institute of Software, Chinese Academy of Sciences, Beijing, in 2012. He is an associate research professor at the Institute of Software, Chinese Academy of Sciences, Beijing. His primary research interests include covert channels, security of the mobile OS, and cloud computing.

Yong-Ji Wang received his Ph.D. degree in computer science from the University of Edinburgh, Edinburgh, in 1995. He is a research professor at National Engineering Research Center for Fundamental Software, and State Key Laboratory of Computer Sciences at the Institute of Software, Chinese Academy of Sciences, Beijing. He is the winner of the 2002 One-Hundred-People Program sponsored by the Chinese Academy of Sciences and Chinese State Development Planning Commission for "Introducing Excellent and Competent Overseas Scholars". His work has appeared in more than 150 publications.