Metadata of the chapter that will be visualized in SpringerLink

Book Title	Advances in Intelligent Networking and Collaborative Systems	
Series Title		
Chapter Title	Bio-inspired VM Introspection for Securing Collaboration Platforms	
Copyright Year	2022	
Copyright HolderName	The Author(s), under exclusive license to Springer Nature Switzerland AG	

Corresponding Author	Family Name	**Huseynov**
	Particle	
	Given Name	**Huseyn**
	Prefix	
	Suffix	
	Role	
	Division	Department of Electrical Engineering
	Organization	City University of New York, City College
	Address	New York, NY, USA
	Email	hhuseynov@ccny.cuny.edu
Author	Family Name	**Saadawi**
	Particle	
	Given Name	**Tarek**
	Prefix	
	Suffix	
	Role	
	Division	Department of Electrical Engineering
	Organization	City University of New York, City College
	Address	New York, NY, USA
	Email	saadawi@ccny.cuny.edu
Author	Family Name	**Kourai**
	Particle	
	Given Name	**Kenichi**
	Prefix	
	Suffix	
	Role	
	Division	Department of Computer Science and Networks
	Organization	Kyushu Institute of Technology
	Address	Fukuoka, Japan
	Email	kourai@ksl.ci.kyutech.ac.jp

Abstract

As organizations drastically expand their usage of collaborative systems and multi-user applications during this period of mass remote work, it is crucial to understand and manage the risks that such platforms may introduce. Improperly or carelessly deployed and configured systems hide security threats that can impact not only a single organization, but the whole economy. Cloud-based architecture is used in many collaborative systems, such as audio/video conferencing, collaborative document sharing/editing, distance learning and others. Therefore, it is important to understand that safety risk can be triggered by attacks on

remote servers and confidential information might be compromised. In this paper, we present an AI powered application that aims to constantly introspect multiple virtual servers in order to detect malicious activities based on their anomalous behavior. Once the suspicious process(es) detected, the application in real-time notifies system administrator about the potential threat. Developed software is able to detect user-space based keyloggers, rootkits, process hiding and other intrusion artifacts via agent-less operation, by operating directly from the host machine. Remote memory introspection means no software to install, no notice to malware to evacuate or destroy data. Conducted experiments on more than twenty different types of malicious applications provide evidence of high detection accuracy.

Bio-inspired VM Introspection for Securing Collaboration Platforms

Huseyn Huseynov[1(✉)], Tarek Saadawi[1], and Kenichi Kourai[2]

[1] Department of Electrical Engineering, City University of New York, City College, New York, NY, USA
{hhuseynov,saadawi}@ccny.cuny.edu

[2] Department of Computer Science and Networks, Kyushu Institute of Technology, Fukuoka, Japan
kourai@ksl.ci.kyutech.ac.jp

Abstract. As organizations drastically expand their usage of collaborative systems and multi-user applications during this period of mass remote work, it is crucial to understand and manage the risks that such platforms may introduce. Improperly or carelessly deployed and configured systems hide security threats that can impact not only a single organization, but the whole economy. Cloud-based architecture is used in many collaborative systems, such as audio/video conferencing, collaborative document sharing/editing, distance learning and others. Therefore, it is important to understand that safety risk can be triggered by attacks on remote servers and confidential information might be compromised. In this paper, we present an AI powered application that aims to constantly introspect multiple virtual servers in order to detect malicious activities based on their anomalous behavior. Once the suspicious process(es) detected, the application in real-time notifies system administrator about the potential threat. Developed software is able to detect user-space based keyloggers, rootkits, process hiding and other intrusion artifacts via agent-less operation, by operating directly from the host machine. Remote memory introspection means no software to install, no notice to malware to evacuate or destroy data. Conducted experiments on more than twenty different types of malicious applications provide evidence of high detection accuracy.

AQ1

1 Introduction

Collaborative platforms, groupware, or multi-user applications allow groups of users to communicate and manage common tasks. Many companies, industrial infrastructures, government agencies and universities rely on such applications periodically. All of these systems contain information and resources with different degrees of sensitivity. The applications deployed in such systems create, manipulate, and provide access to a variety of protected information and resources.

Balancing the competing goals of collaboration and security is difficult because interaction in collaborative systems is targeted towards making people, information, and resources available to all who need it, whereas information

L. Barolli et al. (Eds.): INCoS 2021, LNNS 312, pp. 1–10, 2022.
https://doi.org/10.1007/978-3-030-84910-8_4

security seeks to ensure the availability, confidentiality, and integrity of these elements while providing it only to those with proper authorization. Protection of contextual information and resources in such systems therefore requires a constant automated mechanism that will address necessary vulnerability points.

Among the several areas of security under consideration for collaborative environments, authorization or access control is particularly important because such systems may offer open access to local desktops or resources through network. In such environments, some applications can gain privilege to access text-based chat, audio/video files, shared whiteboard or any other data. Users need a mechanism not only for identifying collaborators through proper authentication, but to manage files, applications, system processes and so forth. Proposed application aims to eliminate these needs for users. In this paper, we present a single solution to detect malicious applications that tries to surreptitiously gain access to personal files. This solution provide secure environment by constantly checking servers for the presence of keyloggers, rootkits, trojans and other malicious applications using cutting edge artificial immune system (AIS) based technology [1,2]. Crucial part of proposed architecture is *KVMonitor* - the virtualization module that collects data (interrupts, system calls, memory writes, network activities, etc.) by introspecting remote servers [3].

The rest of this paper is organized as follows: Sect. 2 provides a brief background on security for collaboration platforms and list of potential threats. Section 3 explains the negative selection algorithm (NSA) and artificial immune system based IDS. Section 4 describes our proposed end-to-end intrusion detection approach for cloud based collaboration platforms. Section 5 provides a detailed performance evaluation of the proposed security approach. Section 6 draws conclusion and discusses future work.

2 Security and Privacy in Collaboration Platforms

Collaboration and communication, hence it is very common for organizations of all sizes to use tools that facilitate connection between their employees. However, with the advancement in technological collaboration platforms, the risk level also goes up. Hence, the people who hold the authority to adopt such platforms must be aware of some hygiene practices to mitigate risks.

Security in collaboration platforms starts with hardening the security of Virtual Machines deployed in the cloud servers. For example, with cloud computing, user data is usually stored in the cloud service providers (CSPs) data centers across the globe, unknown to the user. The security of such data is crucial in any network environment and even more critical in cloud computing, given the fact that files are constantly replicated across different geographical zones. Several possibilities of attacks exist in this realm. One of the most threatening is the *insider attack*, which also considered as one of the largest threats in this decentralized cloud computing environment.

The heterogeneity and diversity of the cloud computing environment opens up a series of avenues for malicious attacks and privacy issues that could constitute a major threat to the entire system. These threats can be classified from three different perspectives: network, application and virtualization.

- **Security threats from a network perspective.** *Denial of Service (DoS) attack* is an age-long threat in various computing and networking areas. DoS creates an artificial scarcity or lack of online and network services. It could happen in the form of distributed denial-of-service (DDoS) or wireless jamming and could be launched on both the virtualization and regular network infrastructures. In case of Software Defined Networks (SDN), DoS attacks have limited scope, as described in [4], DoS attack on the network edge will affect only the attacked vicinity and not the entire network. Therefore, due to the autonomous and semi-autonomous nature of edge data centers, the attack might not lead to a complete disruption of the core network infrastructure. Another known network-based attack technique is *Man-in-the-Middle (MitM)*, characterized by the presence of a third malicious party interposed between two or more communication parties and secretly relaying or altering the communication between such parties. The potency of an MitM attack on mobile networks has been proven in various works and literature [5,6]. Such attacks would be even more threatening for the SDN scenario, considering that SDN heavily relies on virtualization, hence launching an MitM attack on multiple VMs could very easily affect all other elements on both sides of the attack (Fig. 1). AQ2

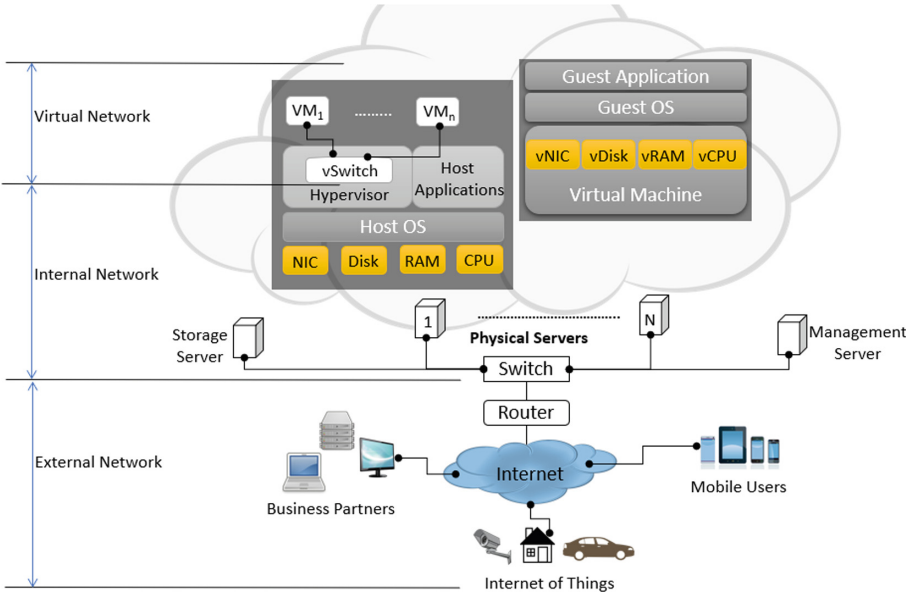

Fig. 1. Network layers in cloud computing infrastructure

- **Security threats from system and application perspective.** Third party applications running in remote servers can pose a fatal security threats by exposing virtual machines to different malicious applications. When virtualization software such as hypervisor or container engine is attacked, remote applications can fail and data can be leaked. Interim attacks through manipulated or malware-infected remote applications or spread of infection to other cloud-based software and data leakage can occur. *Keyloggers*, *rootkits*, *spyware*, *adware*, *ransomware*, *worms*, *trojans* and other nefarious threats are considered as potential risk factors for virtual machines. Exploitation of vulnerabilities in open SDN systems can occur, also known as *hyperjacking*, in which a hacker takes control over the hypervisor that creates the virtual environment within a VM host.
- **Security threats from a virtualization perspective.** While virtual machines are relatively secure because they provide a completely isolated computing environment, containers are vulnerable since they share a single operating system. One of the possible threats in SDN is *VM manipulation*, which mainly affects the virtualization infrastructure. The adversary in VM manipulation is mostly a malicious insider with enough privileges or a VM that has escalated privileges. In addition, arbitrary container access manipulation can lead to a control takeover attack on the container, and there is a possibility of data manipulation or data leakage through open API vulnerabilities in cloud-based applications.

Proposed work mainly lies on detecting security threats in Virtual Machines within system and application perspective. Designed approach employs artificial immune system (AIS) based algorithm for anomaly detection. One significant feature of the theory immunology is the ability to adapt to changing environment and dynamically learning. AIS is inspired by the human immune system (HIS), which has the ability to distinguish internal cells and molecules of the body against diseases [1].

3 Artificial Immune System Based Intrusion Detection

Anomaly-based intrusion detection system monitors network traffic and user/system activity for abnormal behavior. Unlike the signature-based detection method, the anomaly-based IDS can detect both known and unknown (zero-day) attacks. Hence, it is a better solution than the signature-based detection technique if its system is well designed [4]. Therefore, efficiency of anomaly-based IDS depends on multiple requirements such as what kind of algorithms have been deployed, what is the main target, understanding generated input data, application run-time and so on.

Artificial Immune System (AIS) is a type of "adaptive systems", inspired by theoretical immunology and observed immune functions, principles, and models, which are applied to problem-solving. Immunology uses models for understanding the structure and function of the immune system. Simplification of such biological immune system models can produce AIS models that when applied

to determined problems, could be the basis of artificial immune system algorithms and consequently computer programs [1]. An important mechanism of the adaptive immune system is the "self/nonself recognition". The self-nonself (SNS) model is an immunology model that has been successfully utilized in AIS in the design of IDS systems to detect malicious activities and network attacks in a given operating system. Immune system is able to recognize which cells are its own (self) and which are foreign (nonself); thus, it is able to build its defense against the attacker instead of self-destructing [2].

3.1 Negative Selection Algorithm

The negative selection algorithm is based on the self-nonself model of the human immune system (HIS). The first step of the NSA according to Forest et al. [7] involves randomly generating detectors (which is the AIS's equivalent of B cell in HIS) in the complementary space (i.e., space which contains no seen self elements) and then to apply these detectors to classify new (unseen) data as self (no data manipulation) or nonself (data manipulation). Several variations of NSAs have been proposed after the original version was introduced (Forest et al., 1994); however, the main features of the original algorithm still remain. The whole shape-space U is divided into a self set S and a nonself set N with

$$U = S \cup N \; and \; S \cap N = \emptyset \tag{1}$$

There are two steps or phases in NSA, known as *detector generation* phase and *nonself detection* phase. In the first step, a set of detectors is generated by some randomized process that uses a collection of self as the input. Candidate detectors that match any of the self-samples are eliminated, whereas unmatched ones are kept [2]. Algorithm 1 shows a pseudocode of a basic negative selection algorithm. At the detector generation phase, normal profiles (also called self profiles or self samples) which have been extracted from the training data are used to generate random detectors. Each data instance in the normal profile is obtained from the data instances captured by the system during periods of normal activity (i.e., during the absence of any malicious applications). A detector is defined as $d = (C, r_d)$, where $C = \{c_1, c_2, ..., c_m\}, c_i \in \mathbb{R}$, is an m-dimensional point that corresponds to the center of a unit hyper-sphere with $r_d \in \mathbb{R}$ as its unit radius. For the generic NSA shown in Algorithm 1, $r_d = r_s$ [8].

Algorithm 1. A Generic Negative Selection Algorithm

```
function GenericNSA(S, T_max, r_s)
    ▷ Where S - set of normal/self profiles, T_max - max. number of detectors, r_s - matching threshold.
    D ← ∅
    while |D| < T_max do
        Generate a random detector (d)
        if d does not match any element in S then
            D ← D ∪ d
        end if
    end while
    for All new incoming samples ν ∈ ∪ do
        if ν matches any element in D then
            Classify ν as a nonself sample
        end if
    end for
    return D
end function
```

Figure 2 shows a basic block-diagram of two NSA phases: detector generation process on the left and nonself detection on the right. Randomly generated candidates that match any self samples are discarded. The detector generation process is halted when the desired number of detectors is obtained. To determine if a detector (C, r_{di}) matches any normal profile, the distance $(dValue)$ between this detector and its nearest self profile neighbor $(X^{normal}, r_s) \in S$ is computed, where X^{normal} is an m-dimensional point $\{x_1, x_2, ..., x_m\}$ and corresponds to the center of a unit hyper-sphere (with r_s as its unit radius). Here d_i is a random candidate detector with center C and radius r_{di}.

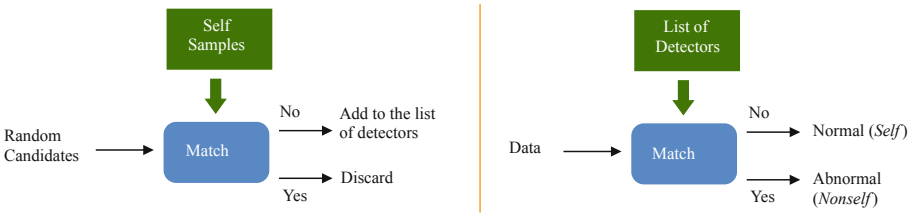

Fig. 2. Detector generation process on the left and nonself detection on the right.

In the proposed work distance *dValue* is obtained using *Squared (Euclidean) distance*, however, depending on architecture, any real valued distance measures can be used (such as *Euclidean distance*, *Manhattan distance*, *Chebyshev distance*, etc.).

$$d(c, x) = \sum_{i=1}^{m} (c_i - x_i)^2 \tag{2}$$

Process of generating random candidates to cover the nonself space employs *Genetic Algorithm*. The self-space consisted of a set S, a subset of $[0,1]^m$; accordingly, a data point was represented as a feature vector $x = (x_1, x_2, ..., x_m)$ in $[0,1]^m$. At the beginning, an initial population of candidate detectors is generated at random. Such detectors then mature through an iterative process. In each iteration, the radius of each detector is calculated as $r_d = dValue - r_s$, where r_s is the variable distance around a self [1,2].

4 Proposed Security Approach

The proposed intrusion detection and mitigation approach, the overall architecture of which is depicted in Fig. 3, provides security in cloud-based networks by automated, intelligent analysis of network flows and system level forensics, followed by mitigation actions being taken in accordance with the decision of IDS component. *KVMonitor* is the crucial part of nonself detection phase and provides an API for translating a virtual address to a physical one [3]. To introspect a virtual disk with the qcow2 format, *KVMonitor* uses network block device (NBD) for QEMU. By doing so, it allocates a real disk space only to used blocks, therefore saving a disk space. Several conducted experiments confirmed efficiency of memory introspection using *KVMonitor* [3].

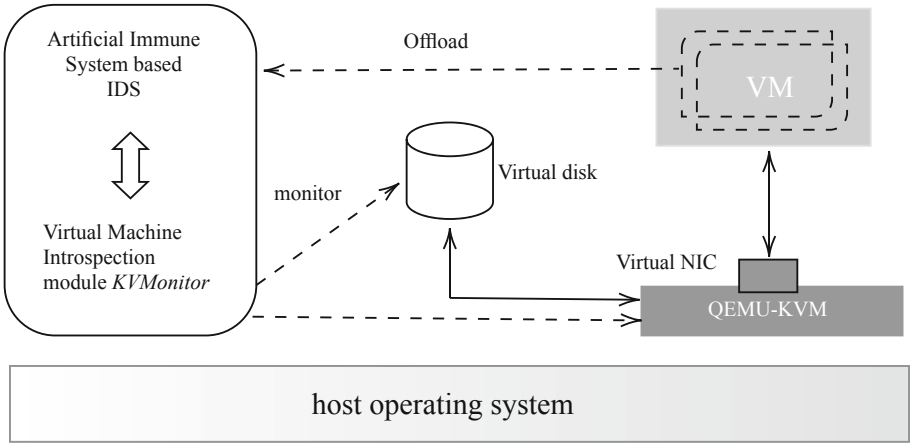

Fig. 3. Basic architecture of proposed intrusion detection system.

List of detectors obtained from the detector generation phase is being used in the second phase. During the nonself detection phase (Fig. 2), *KVMonitor* constantly introspects multiple VMs and returns raw feature values to the IDS. Next, the application converts these features into the binary tuples and begins matching process. If application finds a match for any incoming set of features among the detectors, it is immediately notifies administrator about potential anomaly. Primary focus is made on the following features:

- **Keyboard Driver**: *XkbGetState()*, *XKeysymToString()*, *XkbRules()*.
- **Memory Usage**: system calls *Read()* and *Write()*, *RssFile()*, *RssShmem()*.
- **File System**: *ReadFile* and *WriteFile*, *CreateFile*, *OpenFile*.
- **Network Flow**: *Send*, *Sendto*, *Sendmsg*, *TCP_socket*, *UDP_socket*.

The controller at IDS periodically collects these entries from virtual machines, which are retrieved by the *KVMonitor* at regular intervals. Upon retrieval, features are converted into binary tuples for every flow and algorithm begins matching process. While looping over the flow entries, the incoming features are immediately sent to the IDS, without waiting to finish creation of other flow entries. One of the main benefits of AIS based virtual machine introspection is zero load on VM, since IDS operates from the host operating system.

5 Experimental Evaluation

In this section, we provide an experimental evaluation of the proposed security approach using three different types of Linux based keyloggers taken from the open source software list [9]. The experiments were conducted on a host machine with Intel Core i5-11400 @2.60GHz processor and 16 GB RAM. The guest machine was running on Ubuntu 18.04 LTS with allocated 2 GB memory. All malicious applications listed in the Table 1 have been initially installed into the VM.

In order to demonstrate efficiency of proposed system, we have divided experiments into two parts. First, after being logged in to the VM, user starts typing short sentences with periodic pauses (≈40–80 characters). The text can be entered into any application (browser, text editor, etc.) running inside VM (*Chart (a)*). As part of the second experiment, user types long text making certain pauses between sentences (≈400–1500 characters) using any default text editor (*Chart (b)*). We measured time for both scenarios considering fluctuation of several features (keyboard tracking, file access, network flow).

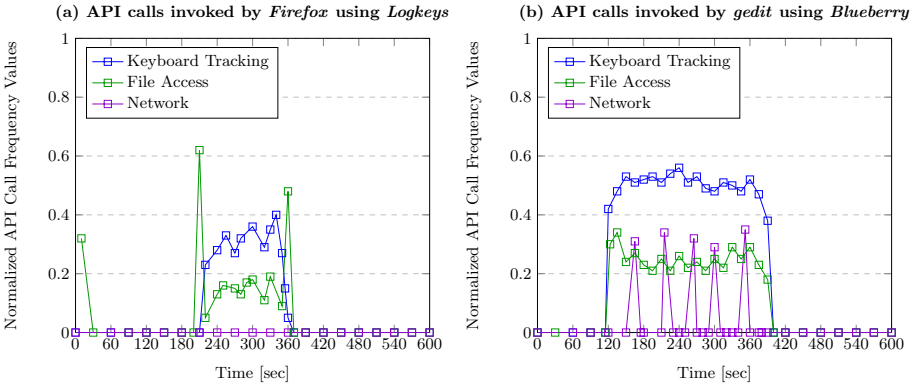

Chart (a) shows the result of anomalous fluctuation of the features depicted by our IDS while typing in the infected VM. The X-axis represents time in seconds and the Y-axis is normalized value of API call frequencies. The normalized API call frequency values are the total value we get during 10 s divided by the maximum value of the whole period (600 s). *Chart (b)* represents second part of the experiment, but with a different keylogger. In this case, keylogger triggers networking features by trying to send captured keystrokes over the TCP protocol to remote server.

Table 1. Three different types of keyloggers used in this experiment

Logkeys	:	Multi functional GNU/Linux keylogger. Logs all entered keystrokes including function keys [10]
Blueberry	:	Opens a stream to the keyboard event handler and gets every key press. Create logs when the buffer gets 300 characters and sends it to the remote server over TCP protocol [11]
EKeylogger	:	Sends recorded keystrokes every 10 sec using SMTP protocol [12]

Proposed IDS was able to detect all listed keyloggers within the first 10 s of their launch. This time is allocated as an interval for VM introspection and can be reduced depending on IDS configuration. Moreover, application efficiently detects other types of malicious applications (trojans, rootkits, adware and so on) without human interaction. Artificial Immune System based IDS is able to track minor deviations from normal profile triggered by malevolent processes. Conducted experiments on more than twenty different malicious application demonstrate high detection accuracy and efficient VMI without any user being engaged. The proposed security approach is promising for achieving real-time, highly accurate detection and mitigation of attacks in cloud-based servers, which will be in widespread use in the 5G and beyond era.

6 Conclusions

Collaboration solutions have become key to enabling remote work, and if the proper steps are taken to securely configure and deploy them, the risks they introduce can be mitigated. As these platforms become used more heavily in regular business, it is increasingly imperative that organizations have threat intelligence feeds in place, and vulnerabilities impacting these platforms are identified and addressed promptly. In this paper, we provided a distributed solution to secure cloud-based servers for collaboration platforms. We began by examining and classifying potential vulnerabilities for such systems. Next, we presented Artificial Immune System algorithm that is used in proposed application. Following by describing overall IDS architecture and providing experimental evaluation

of presented application. Our future work will include an extension of current introspection by accessing virtual machines remotely. Initial experiments were successfully conducted to introspect a virtual machine over the GRE tunnel. Continuous tests on many different malicious applications provide capability to detect large attack surface in a variety of network structures. We believe that our study helps to introduce a new model for securing collaboration platforms and provide best practices on issues that has high impact on security and privacy.

References

1. Igbe, O., Saadawi, T., Darwish, I.: Digital Immune System for Intrusion Detection on Data Processing Systems and Networks, March 2020. Patent No. US 10,609,057; Filed, 26 June 2017; Issued 31 March 2020
2. Dasgupta, D., Nino, F.: Immunological Computation: Theory and Applications, 1st edn. Auerbach Publications, Boca Raton (2008)
3. Kourai, K., Nakamura, K.: Efficient VM introspection in KVM and performance comparison with Xen. In: Proceedings of the 2014 IEEE 20th Pacific Rim International Symposium on Dependable Computing, PRDC 2014, pp. 192–202, USA. IEEE Computer Society (2014)
4. Roman, R., Lopez, J., Mambo, M.: Mobile edge computing, Fog et al.: a survey and analysis of security threats and challenges. Future Gener. Comput. Syst. **78**, 680–698 (2018)
5. Stojmenovic, I., Wen, S., Huang, X., Luan, H.: An overview of fog computing and its security issues. Concurr. Comput. Pract. Experience **28**(10), 2991–3005 (2016)
6. Zhang, L., Jia, W., Wen, S., Yao, D.: A man-in-the-middle attack on 3G-WLAN interworking. In: Proceedings of the 2010 International Conference on Communications and Mobile Computing, CMC 2010, vol. 01, pp. 121–125, USA. IEEE Computer Society (2010)
7. Forest, S., Perelson, A., Allen, L., Cherukuri, R.: Self-nonself discrimination in a computer. In: Proceedings, Research in Security and Privacy, pp. 202–212, USA. IEEE Computer Society Symposium (1994)
8. Igbe, O., Darwish, I., Saadawi, T.: Distributed network intrusion detection systems: an artificial immune system approach. In: 2016 IEEE First International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE), pp. 101–106 (2016)
9. Top Open Source Keylogger Projects. https://awesomeopensource.com/projects/keylogger. Accessed 2 May 2021
10. Logkeys - a GNU/Linux Keylogger. The source code for the index construction and search is available at https://github.com/kernc/logkeys. Implemented on C and dual licensed under the terms of either GNU GPLv3 or later, or WTFPLv2 or later. Accessed 2 June 2021
11. Blueberry - Simple Open Source Keylogger for Linux. The source code for the index construction and search is available at https://github.com/PRDeving/blueberry. Implemented on C and has open license. Accessed 2 June 2021
12. EKeylogger or simply Keylogger. The source code for the index construction and search is available at https://github.com/aydinnyunus/Keylogger. Implemented on Python for the purpose of testing the security of information systems. Accessed 2 June 2021

Author Queries

Chapter 4

Query Refs.	Details Required	Author's response
AQ1	This is to inform you that corresponding author has been identified as per the information available in the Copyright form.	
AQ2	Please check and confirm the inserted citation of Fig. 1 is correct. If not, please suggest an alternative citation.	

MARKED PROOF

Please correct and return this set

Please use the proof correction marks shown below for all alterations and corrections. If you wish to return your proof by fax you should ensure that all amendments are written clearly in dark ink and are made well within the page margins.

Instruction to printer	*Textual mark*	*Marginal mark*
Leave unchanged	··· under matter to remain	New matter followed by
Insert in text the matter indicated in the margin		or (x2)
Delete	through single character, rule or underline or through all characters to be deleted	or (x2)
Substitute character or substitute part of one or more word(s)	through letter or through characters	new character or new characters
Change to italics	under matter to be changed	
Change to capitals	under matter to be changed	
Change to small capitals	under matter to be changed	
Change to bold type	under matter to be changed	
Change to bold italic	under matter to be changed	
Change to lower case	Encircle matter to be changed	
Change italic to upright type	(As above)	
Change bold to non-bold type	(As above)	
Insert 'superior' character	through character or where required	or under character e.g. or
Insert 'inferior' character	(As above)	over character e.g.
Insert full stop	(As above)	
Insert comma	(As above)	
Insert single quotation marks	(As above)	or and/or or
Insert double quotation marks	(As above)	or and/or or
Insert hyphen	(As above)	
Start new paragraph		
No new paragraph		
Transpose		
Close up	linking characters	
Insert or substitute space between characters or words	through character or where required	
Reduce space between characters or words	between characters or words affected	