Ullrich *et al. EURASIP Journal on Information Security* (2016) 2016:18
DOI 10.1186/s13635-016-0042-3

EURASIP Journal on Information Security

RESEARCH Open Access

The role and security of firewalls in cyber-physical cloud computing

Johanna Ullrich*, Jordan Cropper, Peter Frühwirt and Edgar Weippl

Abstract

Clouds are here to stay, and the same holds for cyber-physical systems—not to forget their combination. In light of these changing paradigms, it is of utter importance to reconsider security as both introduce new challenges. Overcoming the concept of zoned networks, clouds make former internal traffic traveling the Internet. Cyber-physical systems include physical parts into computing and make them potential targets for cyber attacks—a dare as a high number of physical parts have originally been developed to be stand-alone. Cyber-physical cloud computing reinforces the need for a thoughtful security concept. Firewalls are among the basic building blocks in network security and are offered by various cloud providers; however, the question on their quality of protection arises. In this paper, we assess firewall offers of five major cloud providers with respect to cyber-physical system integration. Therefore, we study their default configuration, configuration capabilities, documentation, and filtering behavior. We developed an extendible firewall monitoring tool that enables customers to probe their provider's filtering behavior—an information of interest for risk management or further security consideration. Re-assessing filtering behavior, we found that all offered firewalls have evolved over a time period of more than a year: Configuration possibilities have been enhanced, more illegitimate packets are filtered now, and stateful behavior was discovered at a certain provider.

1 Introduction

Cloud computing has become a standard technology in the business as well as in the consumer sector. Up to 90 % of enterprises are using the cloud in some way [1], and almost everybody is using some sort of cloud application, e. g., Dropbox[1], Spotify[2], or Google Docs[3], in private live. Cloud computing is now a multi-billion dollar market with still high annual growth rates [2], and has changed the way we see computing: It is a utility—similar to water or electricity—now, and companies and individuals are able to flexibly access the desired amount of workload on a pay-per-use basis.

Almost concurrently, the trend towards cyber-physical systems arose. Nomen est omen, such systems interconnect the physical world that we live in with the by now separate world of computing and communication [3]. Based on the motivation that physical objects should become smarter, a number of novel concepts have been proposed, e. g., smart grid, smart production, or smart living. These concepts aim for more efficiency, handling of potential energy shortage, a higher level of sustainability, or simply more convenience.

Obviously, it did not take long time to combine the concepts of cloud computing and cyber-physical systems: There was no reason for cyber-physical systems to resign clouds' flexibility; neither, clouds to refrain from connecting with the physical world especially as the latter were already connected to information technology's communication infrastructures like the Internet.

Security is a major factor in cloud computing as well as in cyber-physical systems; but both concepts suffer from indisposition with respect to security. Cloud computing makes former internal, potentially sensitive traffic travel the Internet as it blurs the traditional concept of zoned networks. Further, responsibilities are shared among the customer and the provider requiring reciprocal coordination. Whereas, cyber-physical systems frequently include physical systems that have originally been developed as stand-alone systems without the vague idea of interconnection. This fact might make them susceptible to even very basic type of attacks. Transferring this traffic as-is over the Internet to a cloud appears to be far from a secure solution.

*Correspondence: jullrich@sba-research.org
SBA Research, Favoritenstraße 16, 1040 Wien, Vienna, Austria

Firewalls have always been a cornerstone of network security as they filter prohibited outgoing and/or incoming traffic and remain even an important appliance within such new concepts. A number of cloud providers offer firewalls to their customers. However, details on their type, location within the architecture, functionality, etc. are rare; documentation is rather unsatisfying due to its recipe style. As a result of lacking knowledge, security considerations are shot in the dark. Aiming to overcome this gap, we investigate firewalls' role and security in cloud computing with respect to cyber-physical systems. Our work is fourfold and represents an extended version of our previous work [4]:

- Cyber-physical systems in clouds is a relatively new concept. Thus, we describe it in detail and highlight already existing as well as potential future scenarios. By the example of these scenarios, we highlight the importance of cloud firewalls.
- We investigate the quality of firewalls at five major cloud providers *Amazon EC2*, *IBM Softlayer*, *Microsoft Azure*, *Google Compute Cloud*, and *Rackspace* as we believe them to be adequate representatives of the public cloud landscape. Therefore, we rent our own instances, investigate the default configuration, the possibilities of configuration, and the relevant documentation. Furthermore, we probe firewall filtering by means of our firewall testing tool.
- The field of cloud computing is changing rapidly. Providers are constantly introducing new functionality or improving current offerings, not omitting firewalls. Thus, we develop a tool that enables cloud customers to check their firewall on their own. We provide this tool on an open-source base to the public. Further, the tool is extendible with additional test scenarios.
- We investigate the change of firewall capabilities at the aforementioned cloud providers over time between the year of 2014 and 2016, again by means of our firewall testing tool. It highlights that all providers offering firewalls further developed their offerings. We identified enhanced configuration possibilities as well as improved filtering. Finally, we investigate additional aspects, i. e., TCP filtering, fragmentation, and Path maximum transmission unit (MTU) discovery.

Section 2 provides background on cloud computing and major providers, firewall types, and cyber-physical systems in general. Section 3 describes the convergence of the two concepts of cloud computing and cyber-physical systems and is followed by Section 4 highlighting respective scenarios. As the providers typically refrain from providing in-depth information on their infrastructure, we develop in Section 5 an extensible tool for firewall probing. In Section 6, we repeat our analysis from our previous work and highlight firewall evolution over time. Section 7 introduces new test cases and focuses on the aspects of TCP filtering, fragmentation, and Path MTU discovery. Section 8 discusses the results and infers practical advice for gaining secure virtual instances. The paper is concluded in Section 10.

2 Background

In this section, we provide background information on cloud computing, its service models, and major cloud computing providers; cyber-physical systems and their advantages in general as well as a typology on firewalls.

2.1 Cloud computing and service models

Cloud computing is defined as "a model for enabling ubiquitous, convenient, on demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." by the *National Institute of Standards and Technology (NIST)* [5]. It encompasses the five characteristics of *on-demand self-service*, *broad network accesses*, *resource pooling*, and *rapid elasticity*. However, clouds appear in various ways and are classified into three service models [5]. The service model assigns responsibilities to customer and cloud provider in different combinations: While with *Software as a Service (SaaS)*, the customer uses a provider's applications running in the cloud by means of a web browser or a certain client; in *Platform as a Service (PaaS)*, a cloud provides a platform including programming languages, libraries, etc. to run customer-created applications. *Infrastructure as a Service (IaaS)* cloud providers however offer resources to the customer where she is able to run her applications including an operating system of choice.

Figure 1 shows a representation of a cloud system consisting of four layers: *infrastructure*, *hypervisor*, *operating system*, and *application*. As can be seen, the lower two layers are always the provider's responsibility, while the upper layers differ. Since in *SaaS* a provider basically offers an application for use, a customer's responsibility for maintenance is very limited; in *PaaS*, the application layer is split into the *platform*, maintained by the provider, and the developed app of the customer. As *IaaS* allows the customer to choose and configure an operating system, the responsibility of maintenance belongs with the customer. In the Infrastructure as a Service (IaaS) model, virtual machines (VMs, often referred to as instances) are paid for by usage, generally in terms of uptime hours, GB of storage and bandwidth used. Users in the IaaS model have control of the whole VM from the kernel layer

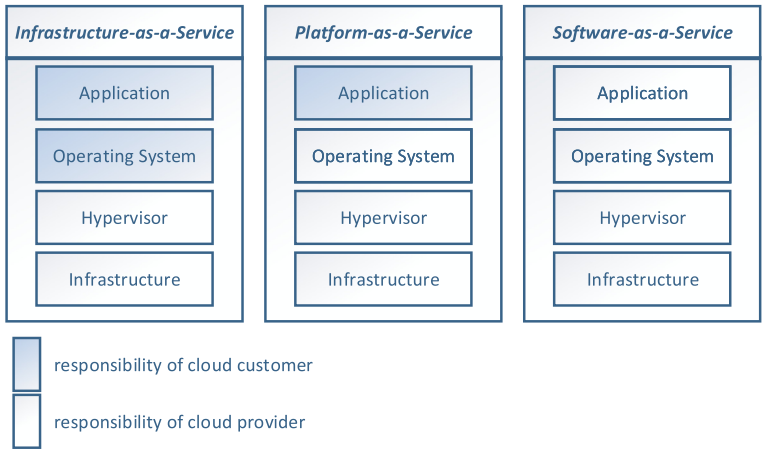

Fig. 1 Responsibility assignment of customer and provider according to service model

upwards: The choice of operating system tends to be left to the customer, although most cloud providers offer a number of pre-specified options for easy installation. The user may then operate the VM as though it was any other remote server, with administration over SSH, hosting relational databases, or providing public facing web services over the internet. Many providers allow for a number of instances to be easily connected inside a group, simulating a LAN in the cloud.

IaaS is the least abstracted level of cloud computing. The next level up, PaaS, offers the customer a slightly more removed but still comparatively customizable cloud computing platform. The operating system is usually set or the choice is much more limited than with IaaS, and the provider offers a number of underlying applications, APIs, or other tools in order to simplify the use of the cloud platform. The definition of PaaS is somewhat broad, and certain PaaS products, such as parts of Microsoft's Azure cloud suite, span more than one category. The final level we see is SaaS. This involves offering a product directly from the cloud, though this does not preclude having certain locally installed elements to add extra features or provide a better service.

The current cloud computing marketplace is a young market and still relatively diverse [6], but a number of larger providers hold considerable market share. This paper looked at a number of different cloud computing providers who all enjoy a considerable level of market share in the cloud computing space, summarized in Table 1. Consistent with the origins and high initial setup costs of a cloud computing platform, they are all large, well-recognized names in the technology sphere. Providers are ranked in approximately decreasing size of market share, although the way in which cloud computing operates makes it extremely difficult to make accurate comparisons. Companies are often under no obligation to report the extent of their revenue which comes from cloud computing, and the use of different metrics makes direct comparison difficult. However, it is universally acknowledged that Amazon is the largest player in the cloud computing space, with as much cloud revenue and capacity as the majority of the competition combined [7]. Many providers offer a number of service types or are difficult to classify. Google's Compute Engine uses an IaaS model much the same as Amazon's Elastic Compute Cloud (EC2), which shares little with Google App Engine, the company's other offering firmly in the PaaS area. Parts of the Azure cloud suite are also PaaS offerings, but their virtual machines are a standard IaaS product.

2.2 Cyber-physical systems and their advantages

In recent decades, physical systems were distinct from computing and communication networks [3]. Such physical systems were the electrical power grid delivering power from plants to customers, cars consisting mainly of mechanical parts, or production systems for manufacturing goods and many more. Although such systems had some basic communication or computing facilities, e. g., ripple control to switch street lights on and off, or engine control units controlling petrol injection, they are considered to be stand-alone. Computing and communication was a system internal. Likewise, computing and

Table 1 A comparison of cloud computing providers in the study

Provider	Service type
Amazon web services	IaaS
IBM (Softlayer)	IaaS/PaaS
Microsoft Azure	IaaS/PaaS
Google	IaaS/PaaS
Rackspace	IaaS

communication systems like telephony or later the Internet were stand-alone. They were intended to work on "immaterial" goods like information or money and had no direct impact on the physical world. The development in the areas of embedded systems, real-time systems, and control however enabled the connection between physical systems and the world of computing and provide new possibilities of monitoring, coordination, and controlling. Such systems spanning among both worlds—the physical world and the cyber world—are called cyber-physical systems [8, 9]. Examples are modern aircrafts full of sophisticated functionality as the autopilot, a body sensor network sensing your vital data for medical purposes, the smart grid being the next-generation power grid, modern factories heavily relying on information technology, modern communicating transportation systems, or fully automated and connected buildings [3, 8]. Cyber-physical systems are multidisciplinary approaches and thus involve a variety of different fields, e. g., mechanical engineering, electrical engineering, medicine and bio-medical engineering, civil engineering, computer science, and telecommunication.

Actual linking between the physical and the cyber world is done by means of sensors and actuators as depicted in Fig. 2. Sensors measure physical quantities and convert them into an electrical signal; this signal is sampled and quantized to make it ready for computing. Calculation is done based on these values before being returned to the physical world by means of actuators that convert electrical signals into a physical action [10].

Advantages of cyber-physical systems are manifold. In the following, we name a few: Physical resources can be accessed remotely from all over the world—similar as communication over the Internet has overcome physical presence of communicators. The combination of physical systems with computing allows to tackle problems of our era, e. g., the exit from nuclear and fossil-fuel energy. Processing immense masses of data allows more efficiency or more convenience.

In the context of cyber-physical systems, also the term *Internet-of-Things* is frequently used. The Internet-of-Things "is the pervasive presence around us of a variety of things or objects—such as [...] sensors, actuators, mobile phones, etc.—which, through unique addressing schemes, are able to interact with each other and cooperate with their neighbors to reach common goals" [11]. As cyber-physical system, Internet-of-Things is an umbrella term for a development that is a convergence of a number of already existing technologies. For the purpose of this paper, we use cyber-physical systems meaning both as the concepts are similar and intersect in large parts.

2.3 Typology of firewalls

Firewalls are among the older but still highly important tools in security and continue to play an important role both in home networks and in enterprises [12]. Firewalls are components residing at the border of two networks and inspect traffic going from one to the other. Modern firewalls are powerful, fully featured security tools capable of matching incoming traffic against complex rules to protect the vast array of modern networks against numerous threats and malicious actors. For best security results, no traffic should bypass the device to guarantee inspection of the whole communication process. Reflecting the historic development of these security tools, firewalls are grouped into three types [13].

The most simple are *packet filters*. They hold a number of rules which define allowed or disallowed traffic; while the first is forwarded to the other network, the latter is dropped. Rules include source and destination address as well as ports identifying the service. The decision is based solely on the currently inspected packet, and no connection context is maintained. The advantages of this general approach allow the firewall to handle a high level of service without any in-depth knowledge; this also allows the integration of newly developed services the firewall is not already aware of. Beyond, the simple architecture has only a limited need for resources in comparison to other types. On the other hand, they do not have full insight into the

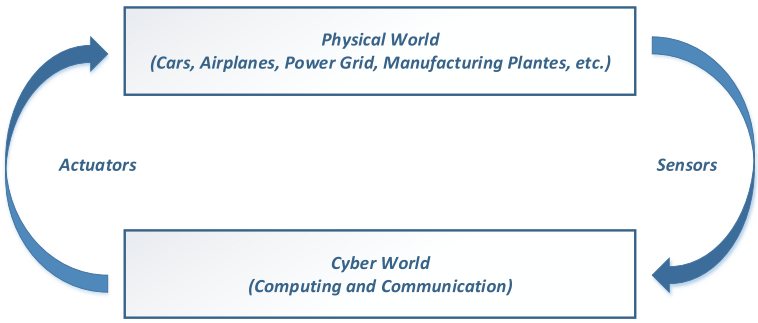

Fig. 2 Cyber-physical systems connect both worlds by means of sensors and actuators

communication as a number of protocols are stateful, i. e., packets may be in general valid, but not at this certain point in communication.

Stateful filters go beyond this approach and maintain a context for connections. This allows the use of previously seen packets as part of the decision regarding the current packet [14]. This way, it is possible to guarantee that certain packet types are protocol-compatible, not only considering the message format, but also considering their point within the communication process, although at the cost of requiring more memory. Typically, real-world stateful filters also use simple packet filtering for the best effect.

The third step in firewall evolution are *application layer filters* which have special code for every application. While this allows even deeper inspections and is thus also considered to be more secure, the drawback is the specialized code for every application. This generally leads to the situations that only the best known services are available, without any extension for niche or novel protocols, and application layer filters are more costly in terms of computing. Currently, *deep packet inspection* is all the rage, which also scans the packets' payloads, e. g., for malware.

For all kinds of firewalls, the permissive and the restrictive approach can be applied. Describing the default behavior, a permissive firewalls allows all traffic to pass unless specified otherwise; restrictive configurations drop everything unless specified. Assuming in general everything as evil, the latter is far more common. The majority of firewalls are able to log events and raise alerts if certain conditions are met.

The fact that an IaaS provider cannot know ahead of time what the client intends to use their machines for means that traditional firewalls are unsuitable, as shown in the comparison of Figs. 3 and 4. In a traditional network, data ingress/egress is restricted and protected by a firewall customized to the specific setup. A cloud network deals with different challenges. The firewalls, provided by the provider, must be capable of protecting individual groups of instances without advance knowledge of the expected traffic. This is not a problem faced by SaaS or (to a slightly more limited extent) by PaaS providers. In SaaS, the provider knows exactly the needs and requirements of the application and can tailor the security set up accordingly. The situation for PaaS is slightly more complex given that the provider does not have complete knowledge of the use of the platform, but still knows what mechanisms are supplied for the client to use. As a result, the most interesting case is IaaS, as here the firewall providers must adapt the most to the changes brought by the cloud environment; flexibility may be one of cloud computing's most heralded advantages, but it may also present problems when we consider the security of the cloud environment. After all, securing any border without knowing in advance what is to be expected is considerably harder than when one has exact knowledge, as in SaaS, of what should be allowed in or out.

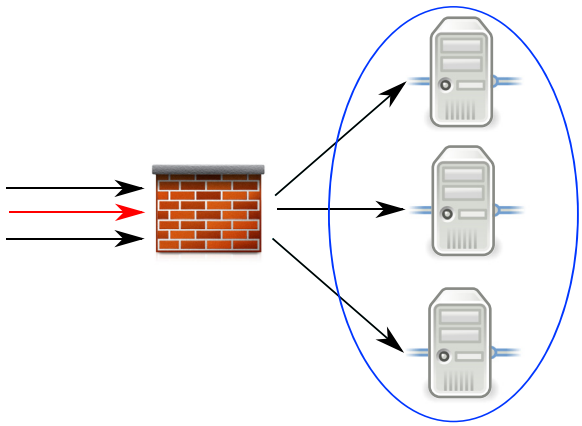

Fig. 3 Simplified diagram of a traditional network: data ingress/egress is restricted and protected by a firewall customized to the specific setup

3 Cyber-physical cloud computing

At the moment, we experience two new computing paradigms. On the one hand, there is cloud computing. Its major improvement is the provision of immense computing resources for an acceptable price and with high flexibility as well as scalability; but clouds are restricted to the virtual world of computing. On the other hand, there are cyber-physical systems that allow to access the physical world from the cyber world. Their drawback however might be their more limited amount of computing resources. Thus, it was not a long time before these two paradigms were merged—especially seeing the fact that

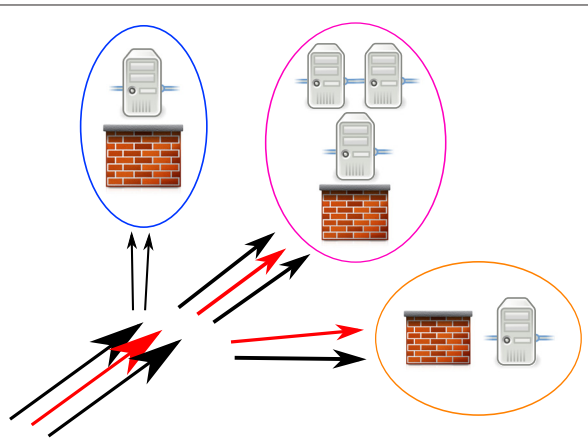

Fig. 4 A cloud network deals with different challenges. The firewalls, provided by the provider, must be capable of protecting individual groups of instances without advance knowledge of the expected traffic

both cloud computing as well as cyber-physical systems were capable of interconnection.

Being already a subject of *NIST*, cyber-physical cloud computing is defined as "a system environment that can rapidly build, modify, and provision cyber-physical systems composed of a set of cloud computing based sensor, processing, control, and data services" [10]. Their benefits are explained as the following [10]:

- Cyber-physical clouds lead to more efficient resource use as it overcomes the rigid structures of today's systems that are specialized for a certain purpose. Various of these specialized systems own their equipment for the same goal, e. g., measuring a certain quantity every now and then or influencing a certain parameter once in a while. With cyber-physical clouds, components like sensors can be reused for various purposes and are thus not underutilized anymore.
- (Physical) functionality is offered as a service to customers, and this modularity allows to create new systems or modify existing systems within a short time and reasonable amount of work.
- Like advanced programmable interfaces (API) for configuration in clouds [5], there will be a framework allowing easy (self-)configuration making it simple to create oneself a cyber-physical cloud.
- Devices are aware of the context that they are in, e. g., whether they are measuring in a residential home, a supermarket, or a production facility. Due to this additional environmental information, gained data appears more valuable for processing and might be more fertile for exploitation in autonomous systems.
- The provision of easily accessible and configurable resources allows to adapt the system to prevent or recover from a failure—an advantage for availability and resilience.

Combinations of cyber-physical systems and cloud computing are already available. By now, however, not all benefits are available as realization is in its early stages. Nevertheless, we are able to see the potential of such an approach.

4 Scenarios combining cyber-physical systems and the cloud

Cyber-physical systems connect the physical world with the world of computing and are rather diverse in their appearance. Their physical part might not only be as small as a single sensor but also assume large proportions like a whole factory or power grid. Likewise, the share and role of (cloud) computing varies from rather simple data representation to complex planning of efficient production. This section provides a selected blend of such scenarios that appear to be representative for cyber-physical cloud computing. Some of these scenarios are already daily business, some are just brought to the market, and others are still visionary and will take some time to become reality. At the end, we draw a conclusion on these scenarios and highlight their need for security in general and for firewalls in particular.

4.1 Ubiquitous fitness and healthcare

Smart phones are carried with their owner (almost) everywhere, and their sensors enable ubiquitous measurement of the world that we are living. This data gives valuables clues about our daily behavior and subsequently to our personal constitution. Exploited for the good, your mobile is able to become your personal fitness trainer as realized for example in the *runtastic*[4] app. Tracking your jogging exercises, it decides whether you meet your fitness goals or whether you improve. This data is not only then transferred to a public or private cloud [15] as it provides more computing capabilities for data analysis than the smart phone but also allows to compare with other users for the sake of competition and motivation.

While smart phones are strictly speaking computing and no "physical" devices in the sense of this paper's definition of cyber-physical systems, this is not yet cyber-physical cloud computing. However, sensing with your smart watch, a wristband as sold by *Jawbone*[5] or necklace, e. g., *Bellabeat Leaf*[6], makes it cyber-physical. Such devices further allow accurate measurement of vital parameters like pulse or respiration rate and skin temperature. Such permanent measurements might help asthmatics to be alerted before a shock or trigger emergency calls in case of abnormalities. Beyond, implantable chips might give you urgent drugs in the future [16].

4.2 Interactive architecture

People have to tell a house their needs, e. g., by pressing the switch in case of desire for light. Conveniently, the house would infer its residents desires automatically without such direct interactions. Interactive architecture aims for that, i. e., an "exchange of information with the user, in such a way that the interactive system adjusts its assumptions about the user's needs and desires." [17] and thus requires a discreet way of monitoring people. Wearables as smart watches, wristbands, or necklaces could be used therefore and allow the house to open the door for you or dim the light. Likewise, the cloud appears appropriate due to its computing power and flexibility [17].

4.3 Vehicle updating

By now, cars have been stand-alone physical systems; modifications—even those in software—have required visiting the garage. Appearing inadequate for our modern, fast-living world, car manufacturers now envision

automatic updates similar to those in smart phones. Software updates and new features should be delivered over the Internet—independently of a car's location and without any connection to dedicated tools of the manufacturer—creating a cyber-physical system. *Tesla* has been the first manufacturer using over-the-air updates [18] with others like *BMW*, *Ford*, or *Volvo* to follow. As the computing demands for delivering the updates are hard to forecast, cloud flexibility and scalability appear to be suitable. The manufacturer *Ford* uses the public *Azure* cloud, while *Volvo*'s platform is hosted in an *Ericsson* cloud [19].

4.4 Cloud-based manufacturing

Clouds enhanced computing; now, they continue with manufacturing making factories future-proof. Xu [20] distinguishes two types of how cloud computing is used in manufacturing. *Smart manufacturing* describes the direct use of cloud computing services in manufacturing and are typically centered around their business processing, i. e., replacing their traditional IT infrastructure. Bringing *Enterprise Resource Planning*, *Customer Relationship Management*, *Human Resource Management*, etc., to the cloud supports better collaboration between different sites, wholesalers and retailers, or between customers and producers due to better integration and easier access [20]. Beyond, this way of using the cloud in manufacturing appears to be most promising from today's point of view [21]. A practical example is *salesforce*[7]. Whereas, *cloud manufacturing* describes the move towards service orientation. Distributed physical resources are virtualized and encapsulated before being provided as a service on a central platform to customers [20]—similar to virtualization of computing resources in cloud computing. As a result, manufacturing inherits many of a cloud's inherent characteristics like multi-tenancy, resource pooling, pay-per-use, or self-service [22]. In contrast to smart manufacturing that is becoming state-of-the-art in industry at the moment, cloud manufacturing raises still a number of research questions. Anyway, production systems are highly likely to include certain aspects of cloud computing.

4.5 Virtual vehicles

Virtual vehicles are a concept for information acquisition, i. e., gathering information for search, tracking, or even surveillance purposes. Instead of sending a vehicle for every single operation, the cloud concept is reused and a number of virtual vehicles share a single physical vehicle and all its sensors [23]. The authors present this concept by the example of a quadrotor helicopter. This concept is similar to cloud computing and manufacturing insofar as physical resources are shared; but in addition, objects are moving in space. Being a potential concept of the future, there are a number of open research questions. For example, virtualization would not only have to schedule computing resources but also to decide the trajectory of moving.

All these scenarios require heavy networking: Vital data or data about your residental behavior crosses the Internet a couple of time, i. e., rather private data is available in the generally accessible Internet. Car updates that potentially change a car's behavior transit the Internet as well; likewise, data possibly changing or interrupting a production plant and also virtual vehicles are controlled via a network. In conclusion, the following holds for all scenarios:

1. Data is traveling from a source to a sink over the Internet.
2. This data is rather sensitive with regard to privacy or security.

Thus, network protection is of utter importance. Firewalls are the most basic building block therefore, and their role and guaranteed level of protection has to be understood in detail.

5 Firewall testing tool

In this section, we present our firewall test tools. First, architectural aspects are considered; then, we describe our implementation.

5.1 Architectural consideration

Although firewalls are offered by cloud providers, none of them allows deep insights into the architecture. The available documentation as indicated later in Table 2 is typically of limited use due to being more like a handbook. From a customer's perspective, the following is known: (1) We are able to configure ports and addresses, mostly from a web interface for a group or a single instance. (2) Every customer is able to make a configuration for her own needs. However, the providers refrain from stating whether the customer can configure partially a perimeter firewall this way or a hypervisor-based firewall. While Amazon's approach of security groups lets one think of a more de-centralized approach like the second alternative, Microsoft Azure naming of *end points* lets one rather believe in a configurable perimeter. In the case of a hypervisor-based solution, we are further not able to infer whether the firewall is placed in front of the hypervisor or afterwards directly before the guest operating system; see for both alternatives Fig. 5.

While this seems minor at first sight, the order may have serious impact: Hypervisors are not obliged to deliver packets onto the guests without any alteration. For example, they can decide to reassemble fragments before forwarding due to performance issues. One has to be aware that the connection between hypervisor and guest is not constricted to typical network requirements like a

Table 2 Default settings of firewalls from major cloud service providers

Provider	Firewall available	Web interface	Documentation available	Inbound/outbound separated	Configuration parameters	Default inbound config	Default outbound config
Amazon EC2	✓[8]	✓	✓	✓	Transport layer protocol[9], port (single or range), source (anywhere or single IP)	SSH (TCP, 22) from anywhere[10])	* to anywhere
Google Compute Engine	✓[11]	✓	✓	✗[12]	Transport layer protocol, port (single or range), source (anywhere, single IP or range)	ICMP, SSH (TCP, 22) from anywhere, RDP (TCP, 3389) from anywhere, UDP and TCP (all ports) from same /16	* to anywhere (not configurable)
IBM (Softlayer)	✗						
Microsoft Azure (Classic)	✓[13]	✓	✓	✗[14]	Transport layer protocol, port (single), source (range)	SSH (TCP, 22) from anywhere	* to anywhere (not configurable)
Microsoft Azure (Res. Mgr.)	✓[15]	✓	✓	✓	Priority (of a rule), transport layer protocol, port (single or range), source and destination (anywhere or range), action (allow or deny)	SSH (TCP, 22) from anywhere	* to anywhere
Rackspace	✗						

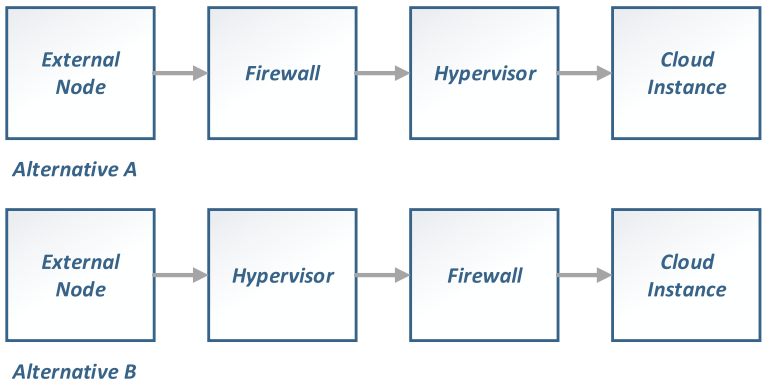

Fig. 5 Alternatives for hypervisor-based firewalls

MTU. Thus, in the first case, the packets investigated by the firewall are how they traveled on the network, while they might have been altered by the hypervisor in the second alternative. This has impacts on the way firewall rules have to be defined as well as the complexity of the firewall mechanism itself, e. g., a hypervisor reassembling the fragments releases the firewall from doing so.

As a consequence of these uncertainties, we have chosen to establish a black box model including the firewall independent of its location in combination with the hypervisor functionality, see Fig. 6. We are able to access the prober as well as the virtual instance for measurements and therefore define the following general test approach:

1. Start of the capturing tool on the receiver, i. e., the virtual instance
2. Establishment of pre-conditions, e. g., TCP handshake
3. Sending of test packet(s) from the prober to the virtual instance
4. and observe whether packet(s) has/have been captured by the sniffing tool

We preferred this over invoking a response from the virtual instance as this would add more chances for failure. By means of this set-up and specific test scenarios, we aim to answer the following question:

5.1.1 Which aspects (e.g., protocols and respective fields) are filtered?

This issue encompasses the ISO/OSI layer the firewall is inspecting and which protocol header field is therefore included. Conceivably, we will heavily work with the network layer protocol IP and transport layer's TCP and UDP. This also include layer-dependent mechanisms, e. g., fragmentation.

5.1.2 Do cloud firewalls reveal stateful or stateless behavior?

Knowledge on firewall behavior enables customers to estimate the extent of protection and the residual measures to gain the required level of protection, e. g., by additional host firewalls. As today's state-of-the-art are stateful firewalls, it would be unusual to find plain packet filters. Thus, we will investigate the "extent of statefulness" of the tested implementations.

5.1.3 Are application layer filters implemented?

Application layer filters would imply a large intervention into a customer's traffic and might restrict their free choice of ports. On the other hand, it would provide more control on what is going into the cloud. We limit our research however on application layer filters for HTTP as we believe that this would be the first choice beneath SSH and FTP for providers for implementations, due to the vast number of web servers in the cloud.

Fig. 6 Assumption for black box firewall testing in cloud environments

5.2 Implementation

As a precondition, we presume that at least one port is reachable by the testing client, which is used later as a feedback and communication channel between the test client and the server. Our approach consists of two components: a testing client that executes test cases against a cloud instance and a server component running on the cloud instance and recording the traffic during test execution. A test case implements a concrete scenario outlined in Section 6.2. As a test oracle, the test case can evaluate the network traffic that reached the server instance in order to decide if the test case was blocked by the cloud firewall. In best case, the test case hides a randomized token, e.g., by adding it to the payload, within the test connections and seeks afterwards in the traffic dump for this token. If the token is present, the test case fails because the request has passed the cloud firewall.

Figure 7 gives a detailed view of the communication between the test client and the server component running on the cloud instance. In detail, the client connects to the server component running on a cloud instance using a plain TCP connection. This channel is used for coordination and callbacks. A single test case is stored as a separate file in a directory. The client loads a test case from this directory and creates a test case instance. Afterwards, the client tells the server component over the communication channel to start recording the network traffic using *tcpdump*. Subsequently, the client executes the test case and requests the tcpdump of this session from the server component. The test case instance now evaluates the success of this test iteration using the dump received from the server. The client repeats this steps until all test cases are executed. We released the implementation including all test cases[16].

6 Results: firewall evolution over time

This work extends our previous work on cloud firewalls; see [4]. Public cloud providers constantly improve their offer and might also alter firewall capabilities in this processes. Thus, we reran our experiments in this section, and compare today's results (2016) with past ones from the year of 2014. Thereby, we identify various modifications. In particular, Section 6.1 focuses cloud-based firewalls' default configuration, Section 6.2 explains the utilized test cases for probing using our firewall testing tool, and finally Section 6.3 analyzes the results.

6.1 Default setup at major providers

In this step, we investigate the availability of firewalls and their scope of functionality. Out of the providers in Table 1, Rackspace and IBM (Softlayer) could not be tested as they do not offer a firewall, the subject of this paper. We looked at Amazon's Elastic Compute Cloud (EC2), Microsoft's Azure Cloud Platform, and Google's Compute Engine. It has to be noted though that Azure offers now two deployment models—*Azure Classic* and *Azure Resource Manager*—having impact on our topic. New virtual instances should be launched into the more recent *Resource Manager* mode, while the old instances remain in *Classic* mode if not migrated.

The latter—EC2, Azure, and Compute Engine—provide firewalls to protect instances, groups of instances or virtual instances. However, only Compute Engine names them *firewall*, EC2 goes for *security groups*, and Azure Classic for *endpoints*. Azure Resource Manager additionally provides *network security groups*. Thus, we included them separately into our considerations. All of them are now configurable via the web interface that also allows to instantiate instances; but vastness of the interface makes finding configuration sometimes difficult.

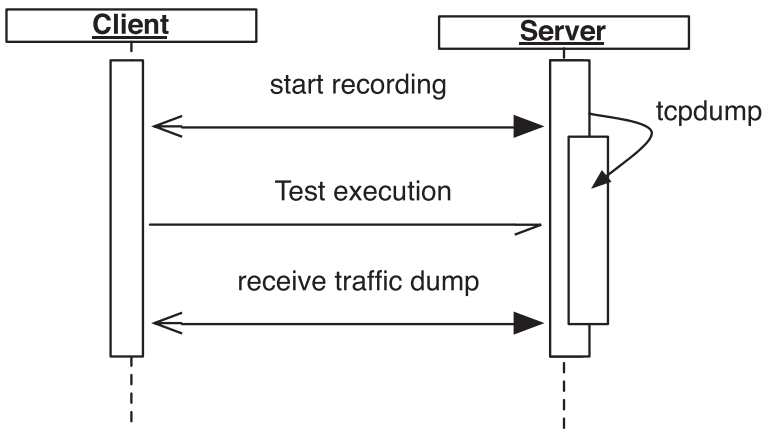

Fig. 7 Communications between the test client and the server component

Adequate documentation is provided, mostly in recipe style and by means of practical examples.

By default, all offer good security by vastly blocking inbound traffic, preventing unsecured instances being exposed to the Internet unintentionally. Nevertheless, there are subtle differences in the inbound configuration as cloud instances have to be remotely accessed anyway: Amazon EC2 and Microsoft Azure solely opens port 22 for SSH in case of Linux instances; alternatively, port 3389 for Remote Desktop Protocol (RDP) in case of Windows instances. Google Compute Engine opens both ports independently of the respective instances. In all cases, rules can be added to allow certain transport layer protocols, ports, and protocols. Although it varies whether single values or (IP/port) ranges are allowed.

Amazon as well as Azure's network security groups provide separate configurations on inbound and outbound traffic, while others provide inbound configuration only and in general all kind of outbound traffic is allowed to pass. Amazon provided only inbound configuration in the older EC2-classic configuration as well, so it might also be included in the future at the other providers. Although outbound traffic is configurable, the default configuration of Amazon and Azure Resource Manager allows any kind of outgoing traffic. Beyond, Amazon is the only provider giving the advice to restrict SSH connections to your own IP address when instantiating a new instance and provides therefore a mechanism for automatic detection.

Azure now not only provides endpoints as a means of protection but also network security groups that contain rules for the protection of a virtual network. While they were only controllable in a programmatic way at the end of 2015, they are now also accessible via the web interface. By default, none network security group is present, and protection is thus solely provided by endpoints. An overall overview can be seen in Table 2.

This paper concentrates on *vendor provided software firewalls*. Some providers, such as IBM Softlayer, offer the option of a dedicated hardware firewall, which is beyond the scope of this paper and has more in common with traditional enterprise networks than the cloud environment. Similarly, we are not looking at local, instance, or OS level firewalls, an area covered by traditional firewall products such as iptables and other commercial offerings. We are primarily interested in the new challenges that the cloud brings to security and how vendor provided perimeter firewalls can work towards mitigating these threats.

Finally, let us conclude the changes since our first measurements in 2014 [4]:

- Amazon EC2 provides functionality to automatically determine one's own IP address—a valuable support for firewall configuration for less experienced users. Amazon's intention is to restrict SSH/RDP access for remote maintenance.
- Google Compute Engine's firewall is now also configurable by means of the web interface. Previously, it was necessary to use Google Cloud SDK via the terminal. This change makes control easier for users without or limited command line experience.
- In its new deployment model, Microsoft Azure, provides network security groups in addition to endpoints. They allow separate inbound/outbound configuration. While they had to be configured by means of the Azure command line in the past, they are now accessible via the web interface.

6.2 Test cases

For our test tool, see Section 5. We implement test cases according to Table 3. With these test cases, we determine the responses of firewalls to specific inputs, in order to classify them and to examine how they would respond to certain well known security threats. The test cases are separated in groups A to E depending on the functionality. Some test scenarios, e. g., 1 or 7, are benign scenarios to test for functionality and connection. For a detailed explanation, we refer to the test cases provided with our testing tool.

We chose to examine the firewalls' responses in the following areas:

A: Internet protocol (IP) Test cases 1 to 6 cover basic aspects of IP and a variety of illegitimate field combinations. These scenarios test the extent of packet investigation on the network layer. At present, we limit the test cases to IPv4 as IPv6 is at the moment not widely supported in public clouds. By now, IBM Softlayer and Rackspace—both not offering firewalls—are the only providers supporting IPv6 natively at their virtual instances.

B: Fragmentation The firewalls' responses to fragmented packets, both normally generated and "malicious" overlapping packet fragments, were measured in test cases 7 to 11. The same operating system was used for all tests, and kernel level measuring tools were employed to ensure the firewall, not the OS, was responsible for the observed behavior. We used both, fragmented ICMP and UDP packets, except for Azure where only UDP was used, as Azure does not permit ICMP traffic at all.

C: Basic TCP and UDP Group C scenarios (test cases 12 to 17) cover basics aspects of the transport-layer protocols TCP and UDP, i. e., invalid source and destination ports as well as invalid checksums.

Table 3 Implemented test cases

ID	Name
A	IP
1	Valid ICMP Request
2	Checksum invalid
3	Invalid packet length
4	Invalid header length
5	Reserved flag unequal zero
6	IP protocol number unequal ICMP, TCP, or UDP
B	Fragmentation
7	Benign fragmentation
8	Overlapping fragmentation
9	Overlapping fragmentation without terminating fragment
10	Overlapping fragmentation in reverse order
11	Tiny fragments
C	Transport layer protocols
12	Invalid source port (TCP)
13	Invalid source port (UDP)
14	Invalid destination port (TCP)
15	Invalid destination port (UDP)
16	Invalid checksum (TCP)
17	Invalid checksum (UDP)
D	TCP flags
18	Null packet (no flags)
19	SYN, FIN
20	SYN, FIN, PSH
21	SYN, FIN, RST
22	SYN, FIN, RST, PSH
23	FIN
E	Stateful behavior
24	SYN in established connection
25	ACK without ACK number
F	Application layer
26	Improper HTTP request

D: TCP flagging Group D (test cases 18 to 23) contains a number of packets with TCP flag combinations, tested without a previously established connection. From these results, we infer whether there are checks on absurd combinations in absence of an established connection.

E: Stateful behavior Stateful behavior, i. e., in combination with an established connection, is tested with test cases of group E (test cases 24 and 25). In general, all stateless test cases should be repeated within a connection. In our first measurements, see [4], we saw at a very early stage that stateless behavior is prevalent, and thus limited the number of developed test cases.

F: Application layer The last group F targets the application layers firewalls. We targeted HTTP as it seems to be one of the most heavily used protocols in clouds.

These areas have been well addressed by traditional firewall products.The absence of any form of firewall logging means that the testing took on a black box approach, with limited information available even with full access to the firewall configuration menus.

6.3 Firewall responses over time

We ran the test cases twice, once at the end of 2014 and now (mid 2016); results of both runs are presented in Table 4. Therefore, we used the firewalls' default configurations with slight adaptions: First, we allowed TCP traffic for our testing tool's synchronization, and second, we opened another port, both for TCP and UDP traffic, for probing. We chose port 100 (or alternatively port 3333) for the first, and port 6666 for the latter. Where possible, we allowed ICMP traffic to pass. In the following paragraphs, we describe our results and especially highlight differences to our results of [4], as they indicate further development of cloud firewalls.

A: Internet Protocol (IP) Firewalls filter all these packets as expected, however, with a single exception. Azure generally disallows ICMP (see test case 1) without any possibility to opt-in. The other providers allow ICMP traffic; Google by default and Amazon offers respective configuration possibilities. The results of these test cases imply that packet investigation on the network layer is not only present but also successfully filters malformed packets. We did not find any differences between our results from 2014 and now. All these test cases were originally written using ICMP; however, Azure's policy forced us to rewrite them (as well as those for fragmentation, see next paragraph) for UDP as a transport layer protocol; we provide both to our readers.

B: Fragmentation While benign fragments (test cases 7) passed the firewall at all providers in 2014, they do not pass in Google and Azure at the moment; unfortunately, we could not identify the exact reasons from the network traces but consider scapy's fragmentation functionality as root. Overlapping fragmentation are mostly filtered; however, the responses differ with overlapping fragments which are terminated (test case 8). While Amazon still

Table 4 Results of test cases per cloud provider (✓ filtered, ✗ unfiltered)

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
Amazon	Q4/14	✗	✓	✓	✓	✓	✓	✗	✗	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	Q3/16	✗	✓	✓	✓	✓	✓	✗	✗	✓	✗	✓	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Google	Q4/14	✗	✓	✓	✓	✓	✓	✗	✗	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	Q3/16	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓
Azure Classic	Q4/14	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	✗	✗	✓	✓	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗
	Q3/16	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗	✓
Azure Res. Mgr.	Q3/16	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗	✓	✓	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗	✓

lets them pass, Google and Azure filters them. Their actions are consistent when the fragments are received in reverse order (test case 10). Looking at the packet captures from the receiver, we saw that fragmented packets are reassembled before reaching the virtual instance, i. e., by the firewall or the hypervisor. We believe that this is done for performance reasons. Changes are prevalent for tiny fragments, i. e., packets that are reassembled from a single fragments: They are now filtered at all providers.

C: Basic TCP and UDP Test cases 12 to 17 indicate checks of transport layer protocol fields. Invalid source ports, i. e., zero, are allowed to pass at all providers; invalid destination ports are filtered. The latter behavior appears obvious when following a port-based filter approach as there is no rule for this port. While invalid checksums were allowed to pass at our first measurements; Google seems to check them now. While transport layer checks were absent in 2014; at least Google seems willing to introduce such checks now.

D: TCP flagging Test cases 18 to 22 test absurd flag combinations, provider reactions differ. While Amazon and Google let them pass, Azure Classic filters all expect the null packet (test case 18). All providers let FIN packets (test case 23) pass albeit there is no established connection; such FINs appear however meaningless without the latter. In conclusion, Amazon and Google do not appear to check flags at all, Azure to a certain extent. Beyond, we did not identify any changes in comparison to our first measurements.

E: Stateful behavior All firewalls allow sending SYN flags in an established connection as well as packets containing the ACK flag without an acknowledgement number, i. e., there are no changes in comparison to our previous measurement. In addition, it appears as there is no stateful behavior at any of the providers.

F: Application layer In our previous measurements, no signs of an application layer firewall were found. Now, it seems that all check for invalid HTTP requests. This might be an indicator that application layer firewalls have been implemented meanwhile. However, we saw that Amazon still accepts the header when sent within an established TCP connection; thus, the results might also be caused by some sort of statefulness.

Additional findings Each platform disallows the use of certain protocols: Google's Compute Engine does not allow SMTP (port 25) or SMTP over SSL (465 and 587); Azure does not permit ICMP packets to be sent or received. In a similar way, EC2 also blocks sending of SMTP mail by default, though this can be enabled by submitting a support request and using a set of Amazon APIs (SES). None of the services responded to either crafted UDP packets or to a UDP scan using the Nmap port scanning tool. The result were the same no matter if the ports were opened or closed in the firewall. We see from this that the firewall does not send an ICMP "Port Unreachable" notification when the port is closed. This is expected for Azure, which completely disallows the use of ICMP, but perhaps less so for Amazon EC2 and Google Compute Engine.

7 Results: an in-depth look on responses

On the one hand, comparison of firewall functionality over time has shown certain improvements; on the other hand, certain functionality has remained unchanged over the last years. Thus, we decided to extend our work and investigate the following issues in more details. First, we focus on overlapping fragmentation. While the firewalls appear to work well at the very first glimpse, there might be more subtle differences and shortcomings, see Section 7.1. Then, we investigate TCP flag-based filtering in more detail, see Section 7.2. Finally, we take a look on unknown TCP options and Path MTU Discovery; see Section 7.3.

7.1 Fragmentation

According to the results of our first measurements, see Section 6.3, overlapping fragmentation appears to be filtered and thus securely handled. In consequence, we aim to take a more detailed look answering the following questions: Are all fragments filtered? Are fragments already assembled when reaching the virtual instance, and how are they assembled? Therefore, we implemented 22 additional test cases according to the ones proposed in [24]; however, translated to IPv4.

In principle, every test case consists of three fragments:

- The first fragment consists of 24 bytes; thereof, the first eight bytes form the ICMP header, or alternatively UDP if ICMP is prohibited. The remainder is a fixed string. Its *More Fragments* (MF) flag is set as there are further fragments underway.
- The third packet has a fragmentation offset of 24, i. e., it is non-overlapping with the first fragment, and could form a legitimate packet with the first fragment. Its MF flag is never set as it is the last fragment.
- The second fragment is sent in between and overlaps with the first and/or the third fragment. Its fragmentation offset as well as its length varies and so does its MF flag. Every test case is first executed with the second fragment's flag unset and then with the flag being set.

The test cases as well as the results are shown in Fig. 8. Every block represents 8 bytes; H an ICMP or UDP header, the remainder letters indicate payload. The columns show the captured packets at Amazon, Google, and Azure Classic instances.

Google and Azure Classic instances show the same behavior: It appears that they drop fragments when detecting overlapping ones; an exception is solely those with their own header having a fragmentation offset of zero. In addition, fragments following such overlapping

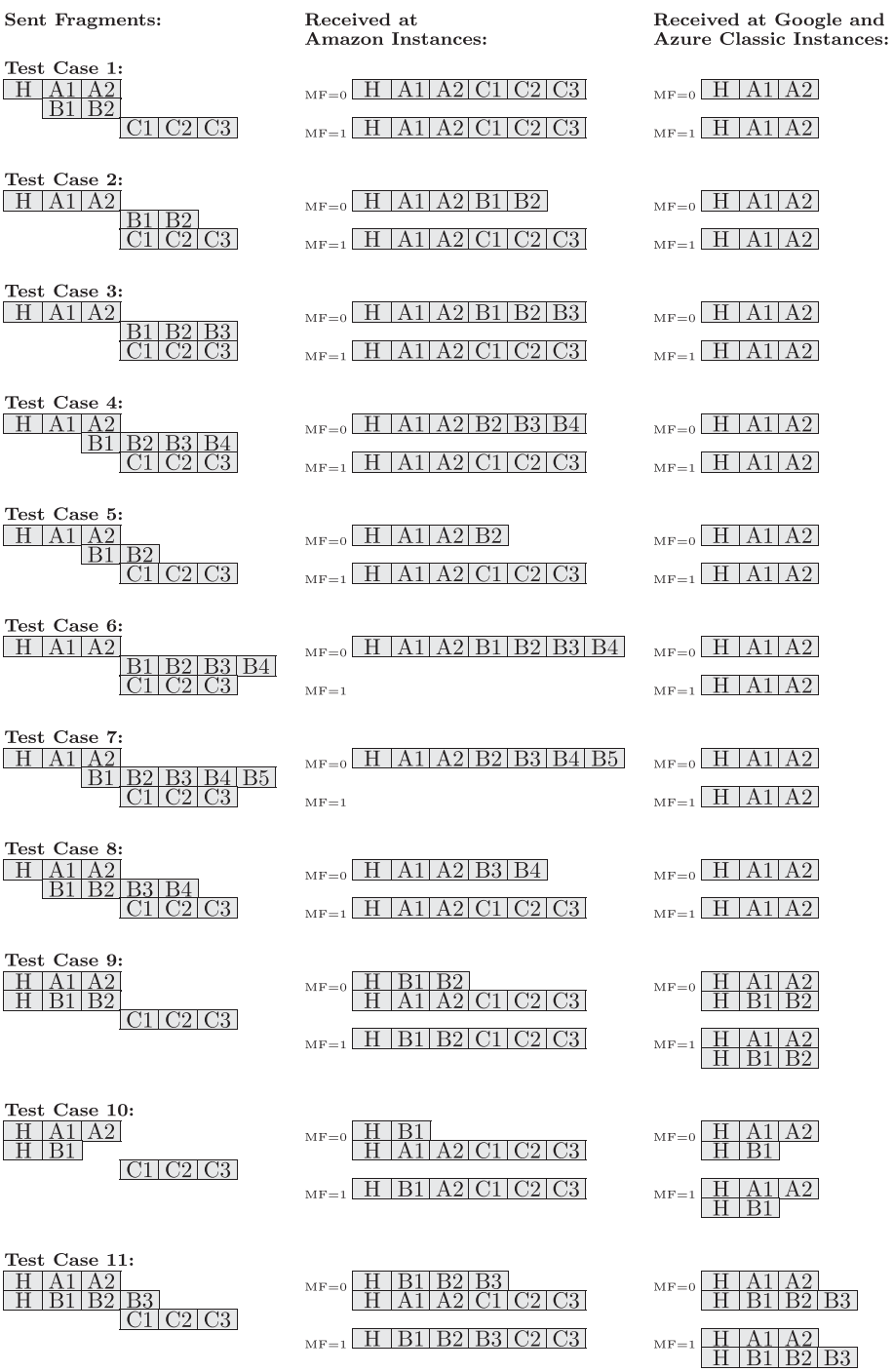

Fig. 8 Fragmentation handling at Amazon EC2, Google Compute Cloud, and Microsoft Azure

fragments are also filtered; the results never show a part of the third fragment. There is further no difference between test cases with a set/unset MF flag. Amazon instances show different behaviors. First, there are two test cases where all fragments are filtered (test cases 6 and 7, each with MF is set). Second, the first fragments appear to be preferred over later ones; these with an unset MF flags are preferred over those with a set flag.

With respect to fragmentation, Azure Resource Manager shows different characteristics than its classic counterpart (not present in the figure). First, it is the only virtual instance where we see (not assembled) fragments in our network captures, i. e., neither the hypervisor nor the firewall seem to reassemble them and deliver them as is to the instance. Second, it always delivers the first and the second fragment; the third is delivered in all use cases except 1 to 8, each with MF unset. In consequence, we assume that all fragments before the first with MF is zero are allowed to pass.

7.2 TCP filtering

Our previous results imply the absence of TCP flag-based filtering at Amazon and Google; only Azure seemed to check to a certain extent. Nonetheless, we aim to expand our knowledge on the exact details of filtering and created 128 additional test cases. In a first step, we probe all flag combinations, i. e., 64 test cases as a consequence of six flags, without a previously established TCP connection. In the ideal case, only packets with a set SYN flags should be allowed to pass in order to establish such a connection. In a second step, we repeat these tests, but establish the connection before by means of a TCP handshake.

The results are shown in Table 5. As predicted, Amazon and Google do not appear to check the flags at all. However, Azure shows not only filtering in the absence of a connection but also stateful filtering behavior—contradicting our forecasts based on our first measurements. In the first case, it filters all except the following four flag combinations (1) S, (2) PS, (3) US, and (4) UPS. This can be seen as a secure configuration. In the latter case, it filters 24 combinations while letting the remainder 40 pass. These filtered cases are combinations of SYN and FIN and/or SYN and RST. These are frequently used for port scanning attempts and are thus of utter importance to be filtered. Some remaining combinations do not make sense in a TCP connection, e. g., the combination of FIN and RST; however, they appear less threatening in comparison to the combinations mentioned above. The provider might thus have chosen to let them pass.

In consequence, we summarize our findings with respect to TCP flags:

- Amazon and Google neither check TCP flags in case of an established TCP connection, nor in the absence of such a connection. This means that each and every TCP flag combination is allowed to pass the firewall.
- Azure shows a more sophisticated behavior in both of its deployment models. Without an established connection, it filters all flag combination except S, PS, US, and UPS.
- In comparison to the results of Section 6.3, we could further discover stateful behavior at Azure. From all (64) flag combination, only 40 are allowed to pass. The filtered appear to prevent various kinds of port scanning.

7.3 Further findings

In the following paragraphs, we discuss three additional findings considering (1) unknown TCP options, (2) Path MTU discovery, and (3) ICMP Echo Requests. First, we aim to send a TCP header with an unknown TCP option. The test case included the option type 111 (which is undefined) and 10 bytes of payload. Our results show that these option could pass at all tested providers.

Second, Path MTU discovery searches for the maximum transmission unit that is able to pass a network and is known for causing "black holes" [25] in case of misbehaving firewalls. If the transmitted bytes exceed a network's MTU, a router en-route fragments the latter. However, packets with the DF (don't fragment) bit set are not allowed to be fragmented en-route, and thus an ICMP destination unreachable (ICMP code 4)/fragmentation needed (ICMP type = 4) is returned. If a firewall now filters exactly these ICMP message, it breaks Path MTU discovery and creates a black hole for packets. In consequence, firewalls that allow outgoing IP packets with a set DF bit must also allow this certain kind of ICMP message as a response. We performed manual checks leading to the following results: Amazon allows outgoing packets with DF is one as well as incoming ICMP destination unreachable messages; Google filters all kind of ICMP messages with type is three (but letting others pass!), and both deployment models of Azure appear to block all kind of ICMP messages, see also Section 6).

During this checks, we found another issue of interest. While Azure Resource Managers prohibits Echo Requests from the virtual instance to other targets; Azure Classics seems to let them pass as well as their replies. This implies that there are some statefulness insofar as echo replies to outgoing requests are allowed to pass. In consequence, we re-tried our checks at Amazon Classic for Path MTU discovery answering to a previously outgoing UDP packets, however, still without success. In consequence, only Amazon shows behavior that is accordant with RFC 2979 [25].

Table 5 TCP filtering behavior with and without an established TCP connection

	Amazon and Google		Azure (both)	
Connection est.	No	Yes	No	Yes
(null)	✗	✗	✓	✗
F	✗	✗	✓	✗
S	✗	✗	✗	✗
SF	✗	✗	✓	✓
R	✗	✗	✓	✗
RF	✗	✗	✓	✗
RS	✗	✗	✓	✓
RSF	✗	✗	✓	✓
P	✗	✗	✓	✗
PF	✗	✗	✓	✗
PS	✗	✗	✗	✗
PSF	✗	✗	✓	✓
PR	✗	✗	✓	✗
PRF	✗	✗	✓	✗
PRS	✗	✗	✓	✓
PRSF	✗	✗	✓	✓
A	✗	✗	✓	✗
AF	✗	✗	✓	✗
AS	✗	✗	✓	✗
ASF	✗	✗	✓	✓
AR	✗	✗	✓	✗
ARF	✗	✗	✓	✗
ARS	✗	✗	✓	✓
ARSF	✗	✗	✓	✓
AP	✗	✗	✓	✗
APF	✗	✗	✓	✗
APS	✗	✗	✓	✗
APSF	✗	✗	✓	✓
APR	✗	✗	✓	✗
APRF	✗	✗	✓	✗
APRS	✗	✗	✓	✓
APRSF	✗	✗	✓	✓
U	✗	✗	✓	✗
UF	✗	✗	✓	✗
US	✗	✗	✗	✗
USF	✗	✗	✓	✓
UR	✗	✗	✓	✗
URF	✗	✗	✓	✗
URS	✗	✗	✓	✓
URSF	✗	✗	✓	✓
UP	✗	✗	✓	✗
UPF	✗	✗	✓	✗
UPS	✗	✗	✗	✗
UPSF	✗	✗	✓	✓
UPR	✗	✗	✓	✗
UPRF	✗	✗	✓	✗
UPRS	✗	✗	✓	✓
UPRSF	✗	✗	✓	✓
UA	✗	✗	✓	✗
UAF	✗	✗	✓	✗
UAS	✗	✗	✓	✗
UASF	✗	✗	✓	✓
UAR	✗	✗	✓	✗
UARF	✗	✗	✓	✗
UARS	✗	✗	✓	✓
UARSF	✗	✗	✓	✓
UAP	✗	✗	✓	✗
UAPF	✗	✗	✓	✗
UAPS	✗	✗	✓	✗
UAPSF	✗	✗	✓	✓
UAPR	✗	✗	✓	✗
UAPRF	✗	✗	✓	✗
UAPRS	✗	✗	✓	✓
UAPRSF	✗	✗	✓	✓

8 Discussion

The cloud providers Amazon Elastic Cloud Compute, Google Compute Engine, and Microsoft Azure have implemented firewalls for protection of their customers' virtual instances. Currently, Azure provides two alternative deployment models Classic and Resource Manager; the latter not only provides endpoints but also (more accurately configurable) network security groups. Current measurements show that quality has increased since our previous measurements at the end of 2014; improvements are found with regard to filtering quality and/or usability of firewall configuration. Differences are especially visible with respect to ICMP filtering, fragmentation, and TCP filtering.

With regard to filtering, Azure Classic as well as Google have changed their fragmentation rules; and tiny fragments are now filtered at all providers. Further, HTTP requests without a previously established TCP connection are not feasible anymore. Beyond, Google now checks TCP and UDP checksums and filters packets with invalid ones. At the first glance, the more recent Azure Resource Manager seems to behave the same way as its older counterpart Azure Classic; however, fragmentation appears to

be handled differently as our more detailed measurements show.

Considering usability, Google Compute Engine caught up and its firewall is now also configurable via the web interface. Previously, the firewall was solely configurable by means of command line instruction and made configuration difficult for less experienced users. We saw a similar change with Azure's network security groups; at the end of 2015, they were just configurable in a programmatic way. Now, they are also easily configurable via Azure web interface.

In comparison to our previous work, see [4], we extended our measurements with respect to three aspects. First, we investigated TCP filtering—both in the absence and presence of an established TCP connection. The results are discouraging as solely Azure provides adequate filtering behavior. In the absence of a connection, just SYN packets (and three of its variants) are allowed to pass. In the other case, it filters all combination of SYN and RST as well as SYN and FIN as they do not make sense in a legitimate connection but are heavily used in port scanning approaches. Amazon and Google neither show some sort of stateful behavior, nor any attempts of filtering absurd flag combinations. We assume that cloud providers deliberately allow a wide range of packets since they do not know in advance what rented instances are used for and thus refrain from the implementation of stateful behavior. Despite being though from a security perspective, cloud providers might prioritize easy handling over better security. As a consequence, we advise customers to additionally configure further means of protection, e. g., a host-based firewall. This firewall could not only perform statefully but also include application layer firewalls and deep packet inspection; however, there is the drawback of additional resources use at the customer's expense.

Fragmentation handling shows a more pleasing picture; but again, strategies vary among different providers. Overlapping fragments are filtered at Google and Azure Classic (and these providers might be more vulnerable to denial-of-service attacks in case an attacker spoofs overlapping fragments); Amazon aims to make meaningful packets from the received packets (and thus might be more vulnerable to fragments overwriting each other). With Amazon, Google, and Azure Classic, it seems that some intermediary, i. e., the firewall or the hypervisor, is reassembling before forwarding traffic to the virtual instance as we could not identify any fragments in our network captures. Our measurements at Azure Resource Manager show distinct (not reassembled) fragments and lets us conclude that there is no reassembling intermediary. The latter means that final reassembly is dependent on the operating system. Path MTU discovery appears to be broken at all providers except Amazon and appears to be a consequence of ICMP blocking. It appears that this protocol is still considered harmful, especially as Azure aims to block it (almost) totally.

In accordance with our previous work, we still believe that logging is one of the larger features missing in today's firewall implementations. As we have not observed any modification in more than a year, we draw the conclusion that a logging feature is not right at the top of the providers' list of priorities. Nevertheless, we claim that logging would be an significant improvement not only for overall security but also for work's quality.

9 Related work

This work is an extended version of our previous work [4]. Related work beyond can be divided into three major areas.

9.1 Firewalls

Firewalls have been used since the early days of the Internet and are well understood. A number of research papers cover aspects of firewalls in traditional network of the pre-cloud era. The scope encompasses a wide variety of topics like firewall design, e. g., [14, 26, 27], or quantitative studies on configuration errors [28, 29] to name a few. Additionally, *Request for Comments (RFCs)* provide guidelines kept in a more practical way; see for example [25, 30].

9.2 General cloud computing security

Most previous work on cloud computing security has focused on the engineering challenges and the business benefits of the cloud [31]. Of the studies which explicitly consider the security implications of the cloud, many papers consider an overview of challenges in cloud security [32, 33], look more at the regulatory and liability issues surrounding moving data into the cloud [31, 34], or focus on theoretical implementations of security infrastructure [35]. Work such as [35] has looked at how firewalls may be adapted for cloud environments, but did not examine any current implementations.

9.3 Cloud identification

There are numerous papers investigating cloud behavior in the wild. While considering the cloud as a black box, they aim to reveal internals on functionality and frequently deal with security issues. Bugiel et al. [36] investigate the security of the Amazon Image ecosystem. Bates et al. [37–40] investigated the impact of resource sharing among multiple virtual instances, especially targeting a shared network interface card (NIC). Bowers et al. [41, 42] consider file duplication for fault-tolerance and local distribution of file storing in clouds. Mulazzani et al. [43] revealed internal behavior of the popular cloud storage solution *Dropbox*.

However, none of them has looked at the firewall implementations of major cloud providers and examines the current state of the art as we did. As far as our research has shown, this is one of the first papers examining the current, in the wild performance of cloud firewalls.

10 Conclusions

The paper at hand examines firewall implementation in public clouds focusing on the major providers Amazon Elastic Cloud Compute, Google Compute Engine, and both deployment models of Microsoft Azure (Classic and Resource Manager). We reran our measurements from our previous work and identify changes in filtering not only behavior but also with respect to usability. Beyond, we extended our measurements and took a more detailed look on TCP filtering, fragmentation, and Path MTU discovery. TCP filtering is solely present at a single provider, Azure even shows stateful behavior, while the remainder do not filter absurd flag combination at all. Fragmentation is handled by filtering overlapping fragments (Google and Azure) or reassembling "most meaningful" packets (Amazon). Finally, it seems that an intermediary, i. e., the firewall or the hypervisor, is already reassembling the packets before forwarding them to the final virtual instance in most cases. We conclude that the offered firewalls provide a solid base of protection for cyber-physical systems and advise customers to configure firewalls according to their needs when running cloud instances and additionally deploy host-based firewalls. We believe that there is still room for further gains: cloud providers could integrate stateful behavior and logging capabilities.

Endnotes

[1]www.dropbox.com.

[2]www.spotify.com.

[3]docs.google.com.

[4]www.runtastic.com.

[5]www.jawbone.com.

[6]www.bellabeat.com.

[7]www.salesforce.com.

[8]Called *security groups*.

[9]Availability of pre-configured rules for popular protocols, i. e., DNS, SMTP, HTTP, etc.

[10]Amazon suggests to change the address to your own IP after instance launching. A mechanism for detecting this IP is provided.

[11]Firewalls are assigned to networks, and not instances.

[12]Configuration of inbound only.

[13]Called *endpoints*.

[14]Configuration of inbound only.

[15]Called *network security groups*.

[16]https://gitlab.sba-research.org/johanna/cloud-firewall-monitoring-tool.

Acknowledgements
This research was funded by P842485 and COMET K1, both FFG - Austrian Research Promotion Agency.

Authors' contributions
JU carried out the main research of this work. JC provided the underlying work of investigating firewalls manually and drafted multiple sections. PF implemented the framework of the firewall test tool and wrote the respective sections. EW actively followed our study and supported in writing as well as revising the manuscript. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Received: 15 January 2016 Accepted: 28 July 2016
Published online: 12 August 2016

References

1. S Florentine, Cloud adoption soars, but integration challenges remain. http://www.cio.com/article/3018156/cloud-computing/cloud-adoption-soars-but-integration-challenges-remain.html. Accessed: 2016-01-06
2. L Columbis, Roundup of cloud computing forecasts and market estimates (2015). http://www.forbes.com/sites/louiscolumbus/2015/01/24/roundup-of-cloud-computing-forecasts-and-market-estimates-2015/. Accessed: 2016-01-06
3. SK Khaitan, JD McCalley, Design techniques and applications of cyberphysical systems: a survey. Syst. J. IEEE. **9**(2), 350–365 (2015). doi:10.1109/JSYST.2014.2322503
4. J Cropper, J Ullrich, P Frühwirt, E Weippl, in *Availability, Reliability and Security (ARES), 2015 10th International Conference On*. The role and security of firewalls in IaaS cloud computing, (2015), pp. 70–79. doi:10.1109/ARES.2015.50
5. P Mell, T Grance, The NIST definition of cloud computing (2011). http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf
6. Cloud computing—the business perspective. Decis. Support Syst. **51**(1), 176–189 (2011). doi:10.1016/j.dss.2010.12.006
7. J Bort, Amazon is crushing IBM, Microsoft and Google. www.businessinsider.com/amazon-cloud-beats-ibm-microsoft-google-2013-11. Accessed: 2014-09-04
8. RR Rajkumar, I Lee, L Sha, J Stankovic, in *Proceedings of the 47th Design Automation Conference*. DAC '10. Cyber-physical systems: the next computing revolution, (2010), pp. 731–736. doi:10.1145/1837274.1837461
9. K-D Kim, PR Kumar, Cyber physical systems: a perspective at the centennial. Proc. IEEE. **100**(Special Centennial Issue), 1287–1308 (2012). doi:10.1109/JPROC.2012.2189792
10. E Simmon, K-S Kim, E Subrahmanian, R Lee, F de Vaulx, Y Murakami, K Zettsu, RD Sriram, A vision of cyber-physical cloud computing for smart networked systems (2013). http://nvlpubs.nist.gov/nistpubs/ir/2013/NIST.IR.7951.pdf
11. L Atzori, A Iera, G Morabito, The internet of things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010). doi:10.1016/j.comnet.2010.05.010
12. K Ingham, S Forrest, A history and survey of network firewalls (2002). https://www.cs.unm.edu/~treport/tr/02-12/firewall.pdf
13. SM Bellovin, WR Cheswick, Network firewalls. Commun. Mag. IEEE. **32**(9), 50–57 (1994). doi:10.1109/35.312843
14. MG Gouda, AX Liu, in *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference On*. A model of stateful firewalls and its properties, (2005), pp. 128–137. doi:10.1109/DSN.2005.9
15. R Buest, Runtastic: A consumer app in the business cloud. http://analystpov.com/cloud-computing/runtastic-a-consumer-app-in-the-business-cloud-23040. Accessed: 2016-01-05
16. N Ravikumar, N Metcalfe, J Ravikumar, R Prasad, Smartphone applications for providing ubiquitous healthcare over cloud with the advent of embeddable implants. Wireless Pers. Commun., 1–8 (2015). doi:10.1007/s11277-015-2999-5

17. H Achten, Closing the loop for interactive architecture. http://cumincad.scix.net/data/works/att/ecaade2015_138.content.pdf. Accessed: 2016-01-05
18. L Mearian, Over-the-air software coming soon to your next car. http://www.computerworld.com/article/2880150/over-the-air-software-coming-soon-to-your-next-car.html. Accessed: 2016-01-05
19. M Murphy, Ford's connected car cloud offering will be powered by Microsoft Azure. http://www.computerworlduk.com/news/infrastructure/ford-powering-its-connected-cars-with-microsoft-azure-3604630/. Accessed: 2016-01-05
20. X Xu, From cloud computing to cloud manufacturing. Robot. Computer-Integrated Manuf. **28**(1), 75–86 (2012)
21. L Columbus, 10 ways cloud computing is revolutionizing manufacturing. http://www.forbes.com/sites/louiscolumbus/2013/05/06/ten-ways-cloud-computing-is-revolutionizing-manufacturing/. Accessed: 2016-01-05
22. D Wu, DW Rosen, L Wang, D Schaefer, Cloud-based manufacturing: old wine in new bottles? Procedia {CIRP}. **17**, 94–99 (2014). doi:10.1016/j.procir.2014.01.035. Variety Management in ManufacturingProceedings of the 47th {CIRP} Conference on Manufacturing Systems
23. SS Craciunas, A Haas, CM Kirsch, H Payer, H Röck, A Rottmann, A Sokolova, R Trummer, J Love, R Sengupta, in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*. HotCloud'10. Information-acquisition-as-a-service for cyber-physical cloud computing, (Berkeley, California, USA, 2010)
24. A Atlasis, Attacking IPv6 implementation using fragmentation. BlackHat Europe (2012). https://media.blackhat.com/bh-eu-12/Atlasis/bh-eu-12-Atlasis-Attacking_IPv6-WP.pdf
25. N Freed, Behaviour of and requirements for Internet firewalls (2000). https://www.ietf.org/rfc/rfc2979.txt
26. MG Gouda, X-YA Liu, in *Distributed Computing Systems, 2004. Proceedings. 24th International Conference On*. Firewall design: consistency, completeness, and compactness, (2004), pp. 320–327. doi:10.1109/ICDCS.2004.1281597
27. AX Liu, MG Gouda, Diverse firewall design. Parallel Distributed Syst. IEEE Trans. **19**(9), 1237–1251 (2008). doi:10.1109/TPDS.2007.70802
28. A Wool, A quantitative study of firewall configuration errors. Computer. **37**(6), 62–67 (2004). doi:10.1109/MC.2004.2
29. A Wool, Trends in firewall configuration errors: measuring the holes in swiss cheese. Internet Comput. IEEE. **14**(4), 58–65 (2010). doi:10.1109/MIC.2010.29
30. D Newman, Bencharking terminology for firewall performance (1999). https://www.ietf.org/rfc/rfc2647.txt
31. M Armbrust, A Fox, R Griffith, AD Joseph, R Katz, A Konwinski, G Lee, D Patterson, A Rabkin, I Stoica, M Zaharia, A view of cloud computing. Commun. ACM. **53**(4), 50–58 (2010). doi:10.1145/1721654.1721672
32. K Curran, S Carlin, Cloud computing security. Int. J. Ambient Comput. Intell. **3**(1), 14–19 (2011). doi:10.4018/jaci.2011010102
33. A Tripathi, A Mishra, in *Signal Processing, Communications and Computing (ICSPCC), 2011 IEEE International Conference On*. Cloud computing security considerations, (2011), pp. 1–5. doi:10.1109/ICSPCC.2011.6061557
34. R Buyya, CS Yeo, S Venugopal, in *High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference On*. Market-oriented cloud computing: vision, hype, and reality for delivering it services as computing utilities, (2008), pp. 5–13. doi:10.1109/HPCC.2008.172
35. S Yu, R Doss, W Zhou, S Guo, in *Communications (ICC), 2013 IEEE International Conference On*. A general cloud firewall framework with dynamic resource allocation, (2013), pp. 1941–1945. doi:10.1109/ICC.2013.6654807
36. S Bugiel, S Nürnberger, T Pöppelmann, A-R Sadeghi, T Schneider, in *18th ACM Conference on Computer and Communications Security*. Amazonia: when elasticity snaps back, (2011), pp. 389–400. doi:10.1145/2046707.2046753
37. A Bates, B Mood, J Pletcher, H Pruse, M Valafar, K Butler, in *ACM Cloud Computing Security Workshop*. Detecting co-residency with active traffic analysis techniques, (2012), pp. 1–12. doi:10.1145/2381913.2381915
38. V Varadarajan, T Kooburat, B Farley, T Ristenpart, MM Swift, in *ACM Conference on Computer and Communications Security*. Resource-freeing attacks: improve your cloud performance (at your neighbor's expense), (2012), pp. 281–292. doi:10.1145/2382196.2382228
39. A Herzberg, H Shulman, J Ullrich, E Weippl, in *ACM Cloud Computing Security Workshop*. Cloudoscopy: services discovery and topology mapping, (2013), pp. 113–122. doi:10.1145/2517488.2517491
40. A Bates, B Mood, J Pletcher, H Pruse, M Valafar, K Butler, On detecting co-resident cloud instances using network flow watermarking techniques. Int. J. Inform. Secur. **13**(2), 171–189 (2014). doi:10.1007/s10207-013-0210-0
41. KD Bowers, M van Dijk, A Juels, A Oprea, RL Rivest, in *18th ACM Conference on Computer and Communications Security*. How to tell if your cloud files are vulnerable to drive crashes, (2011), pp. 501–514. doi:10.1145/2046707.2046766
42. K Benson, R Dowsley, H Shacham, in *3rd ACM Cloud Computing Security Workshop*. Do you know where your cloud files are?, (2011), pp. 73–82. doi:10.1145/2046660.2046677
43. M Mulazzani, S Schrittwieser, M Leithner, M Huber, E Weippl, in *USENIX Security*. Dark clouds on the horizon: using cloud storage as attack vector and online slack space, (Berkeley, California, USA, 2011)