Virtualization for Effective Risk-free Network Security Assessment

http://dx.doi.org/10.3991/ijoe.v10i5.3106

V. Carchiolo, A. Longheu, M. Malgeri, G. Mangioni
Università di Catania, Italy

***Abstract*—Computer networks security is a hard issue, which continuously evolves due to the change of technologies, architectures and algorithms and the growing complexity of architectures and systems. This question is enforced in distributed contexts where the lack of a central authority imposes to set up a proper strategy for both passive and active security. Moreover, it is also essential to test the strategy under as many attack scenarios as possible, to discover and tackle unforeseen situations; this cannot be easily performed on the real system without avoiding either security risks (when it is running) or high costs (if the system must be disconnected from the network during tests). A viable solution is represented by the use of virtualization technologies. Leveraging virtualization permits us to set up an effective and efficient real network duplicate, which can be used for the assessment of security in a non-trivial, risk-free and costs saving fashion.**

***Index Terms*—Virtualization, IT Security, Distributed systems**

I. INTRODUCTION

The security of computer networks is a hard issue continuously evolving over time, due to the change of technologies, architectures and algorithms and the complexity of applications and systems that are growing faster and faster. In particular, in the last decades there has been a growing interest in data protection and security for both users and organizations, while at the same time new scenarios arose, from that of generic distributed systems, to the more specific and somehow fascinating terms as grids [11], cloud computing [7], pervasive and ubiquitous computing [3][21][34]. Moreover, distributed applications running in these contexts make frequent use of agents or mobile agents systems, making the question of security harder and harder. Indeed, mobile agent systems [16] are a special type of software agents that are able to migrate to other hosts, i.e., they transfer their code and context to another host where the execution will continue; due to this nature, companies and users are generally skeptical in allowing uncontrolled code execution, being essentially the same mechanism used by viruses, malware and other malicious codes; this is also one of the main reasons for which the number of commercial applications based on mobile agents is still rather small, despite their potentialities and gained interest [6].

In such a scenario, several security related issues must be considered [12] as cryptography, access control and trust management, intrusion detection, tamper resistance, authentication, privacy and many others [10]. Over the years, a set of procedures, best practices and technologies have been developed to address these issues, in order to protect the resources, people and organizations working in the network. Moreover economic and legal matters related to assessing and managing security has been deeply studied.

Security techniques have been classified in *passive*, based on the use of IDS, and *active*, based on the management of permissions. In most cases, any approach to the security requires to perform a series of tests on the real system to discover and tackle unforeseen situations, thus assessing and validating the endorsed policy. However, performing such tests on a real system however is not a trivial matter, since when it is running and provides access and services to the outside world, the execution of tests could interfere with normal activities even exposing the system to security threats, since tests often imply monitored attack attempts that could be disruptive. On the other hand, the schedule of an offline time period during which tests are performed safely is not advisable due to the high costs of services interruptions, especially for commercial companies.

A viable solution is represented by the use of virtualization technologies [19][28], which aim at improving the utilization of computing resources through their abstraction and re-arrangement; this is accomplished via hardware and software partitioning or aggregation, partial or complete machine simulation, emulation and other methodologies. The concept of virtualization can be applied at different levels, ranging from hardware, operating system, libraries, applications to, more recently, networks and organizations [9][23].

The adoption of virtualization leads to a set of advantages, in addition to the optimized use of computing resources, as the reduction of configuration workload and administrative costs [35], the improvement in application porting [24] and systems survivability [32] the energy saving and the possibility of creating artificial environments for different purposes, as for instance in [14].

In this paper we focus on the last aspect, specifically we propose a tool named ***VirtualNet*** that allows to set up a virtual network, tailored to perform security assessment on such a duplicate of the real network. The evolution of virtual environments indeed guarantees to hold down the amount of both hardware and software resources needed for virtual environment management and, at the same time, to faithfully reproduce the behavior of real systems. Our proposal exploits Xen [2] and Qemu [5] virtualization software to instantiate a large number of virtual hosts that can be configured with identical settings (OS, services etc.) of real counterparts (often already deployed and real offering services to user); moreover, network interfaces of

such hosts can be easily connected to create several virtual networks, allowing us to compose and examine different network configurations.

In summary, leveraging virtualization permits us to set up an effective and efficient duplicate of the real network where the assessment of security can be conducted assuring both its effectiveness, since tests conducted on a virtual network can be identical to those performed in real environments, also in a risk-free fashion, as any (eventually destructive) attack actually does not impact on the real network, finally with a significant costs saving, since the real network is not involved in tests.

The paper is organized as follows. Section II provides an overview about security issues and which tools are used to address such issues. Section III introduces virtualization techniques and tools, whereas in Section IV we present *VirtualNet***,** the tool for network security assessment proposed in this paper, showing some attacks scenarios in Section V. Finally, in Section VI we present our conclusions and future works.

II. IT SECURITY

In IT context, the security is a milestone to preserve any computer network from undesirable malfunctioning and services interruptions [15][29]. Nowadays, security is of critical importance since devices and applications interact each others in an "always connected" distributed world of applications and services

The term *security* refers to several tools and policies aiming at guaranteeing goals as confidentiality, integrity, availability, accountability and others. It can be classified according to different criteria; a first distinction is made between *physical* and *logical* security, being the former an old but still crucial aspect often underestimated (a classical example is the personal password written on a post-it stuck on the office's PC monitor), whereas logical security is sometimes less tangible but it plays a key role in IT. A second classification leads to *passive* and *active* security, in the following briefly discussed together with the corresponding most adopted software solutions.

The *passive* security aims at detecting unusual events or behaviors that might represent an attack; to this purpose, a widely adopted approach is that of Intrusion Detection Systems (IDS), i.e. a combination of hardware and software tools used to reveal unauthorized access (or attempt to access). IDS can be rule-based, when a set of predefined behaviors have been modeled and are used by the IDS to alert whenever one of such rule is matched, or it can be adaptive, being in this case able to detect unforeseen but potentially suspected behaviors. The latter type is better, but it costs more and can run into more false positives. The system monitored by IDS is usually a whole network, where attacks can be issued from different hosts and directed towards many others, concerning an intranet and/or the Internet. The de facto standard Network IDS (NIDS) is the Snort open source software [25], which performs real-time traffic and protocol analysis and packet logging on IP networks; it can be used to detect a variety of attacks and probes through traffic content searching and matching (it indeed comes with a huge database of rules) [26]. In our proposal, Snort has been used as NIDS to detect attacks in virtualization-based scenario representing real environments (see section IV). The active security concerns with tools and techniques intended to prevent attacks, usually by blocking specific traffic patterns. The role of active security is complementary to that of passive, being the former used for prevention and the latter for detection of every malicious behavior that was not recognized (hence not blocked) by active security tools. The main term used in active security is the *firewall*, a hardware/software solution that allows or denies network traffic based on its content or according to specific patterns/policies. The firewall generally is used to protect a network from another, e.g. an intranet from the Internet, and it can work in different modes:

packet level, where the packet content is used to forward or discard it, not paying attention to whether that packet is part of an existing stream;

application level, in which the firewall it can understand certain applications and protocols (as FTP, DNS, or web browsing), and it can detect if a protocol is being abused;

stateful packet inspection, i.e. the firewall tracks all connections passing through it and is able to determine whether a packet is the start of a new connection, a part of an existing one, or is invalid, therefore detecting anomalies within packet streams.

A firewall can be implemented as a software or even hardware solution; in this work, the virtualization naturally leads to software firewalls, in particular we chose the *Iptables* classical stateful Linux firewall working with *Netfilter*, the packet filtering framework inside the Linux 2.4.x and 2.6.x kernel series [20]. In addition, since we exploit virtualization provided by XEN for Linux virtual hosts and Qemu for Windows and FreeBSD virtual hosts (see section IV), we also considered the *pfSense* open source firewall distribution for FreeBSD [4].

Finally, in addition to IDS and firewall, another tool is Nmap [18], an open source port scanner that detects which ports are opened, closed or blocked at a given host. Nmap was used during our security assessment to simulate an attacker that first explores the network to discover unprotected hosts (i.e. with some port opened) to be further used as victims. Nmap is also capable of detecting which services are active on opened ports, which application is offering that service, its version and other details; it also can recognize the OS running on the scanned host by exploiting its OSes fingerprint database, therefore Nmap can be effectively used in security assessment as a powerful tool.

However, the architecture of ***VirtualNet*** allows us to integrate any filter or security devices based on supported operating system or simulated by those systems.

III. VIRTUALIZATION TOOLS AND TECHNIQUES

The first idea of a *virtual* machine is dated at 1960s, when it was introduced to denote a running instance of a physical machine [28]. This was to give the illusion of having a total, direct and exclusive access to that physical machine, actually concealing the need of time- and resource- sharing users had to accept to leverage first expensive mainframe capabilities. With the decreasing of hardware costs and the advent of multiprocessing operating systems, the need for virtualization was drastically reduced during 1970s and 1980s, resuming its importance in the 1990s with the plethora of PC based hardware and operating systems. Further, virtualization has been adopted in more and more disparate contexts, in conjunction with the shifting from centralized to distributed systems,

finally leading to the newest scenarios of virtual storage, virtual networks and virtual organizations.

The adoption of virtualization provides several advantages [24][32][35], one of the most relevant is the optimization in using computing resources, accomplished at different levels, from the multicore CPUs, where manufacturers started to provide native support e.g. with the *Virtualization Technology* from Intel™ or the *Secure Virtual Machine* from AMD™, to OS, libraries and applications. A consequence of optimized use of resources is the costs saving, since virtualization can be effectively used to reduce the hardware acquisition cost, at the same time improving the productivity as many users with different requirements can work simultaneously on the same hardware.

The main three virtualization techniques to allow a generic guest OS to run onto a host system can be summarized as follows:

Emulation, where a complete hardware architecture is replicated via software, thus a guest OS runs on a fully compatible virtual hardware layer. This approach provides great flexibility but with a significant performance overhead.

Native Virtualization, where a software known as virtual machine monitor or hypervisor translates the commands of the guest OS to the host hardware

Paravirtualization, where the guest OS is (partially) modified to allow direct communication with the hypervisor.

This approach requires the presence of proper drivers to achieve guest OS and hypervisor compatibility and consequently the guest OS is aware about the existence of the virtual environment it is running on, whereas emulation and native virtualization do not; conversely, performances are the best available.

Other specific approaches to virtualization exist, as the *Operating System Level Virtualization*, where an OS kernel provides for multiple isolated user-space instances (used for a software to run in isolation from others), the *Resource Virtualization*, in which a given resource of the host system is used by the guest OS (this is the approach adopted in building clusters), and *Application Virtualization*, in which a single application virtual machine allows emulation of a specific environment (for instance, the Java™ virtual machine). In [19], a more detailed description and comparison of several techniques for virtualization can be found.

To implement virtualization, a set of software tools is available, each one based on previously described techniques and coming with its pros and cons. Some of them are XEN [2], Qemu [5], VMware [30], VirtualBox [27] and many others. Here we focus on XEN and Qemu that are both open sources, in particular XEN is a para-virtualization based software that provides great performances, whereas Qemu works in emulator or native virtualization fashion and offers a large hardware support.

Note that it is outside the scope of our work to compare virtualization tools; several papers are present in literature to address this issue, e.g. [1]. Our goal is to exploit virtualization in the ***VirtualNet*** tool for security assessments, where virtual hosts are set up to duplicate real environments; to this purpose, in the following we outline how networking is managed by XEN.

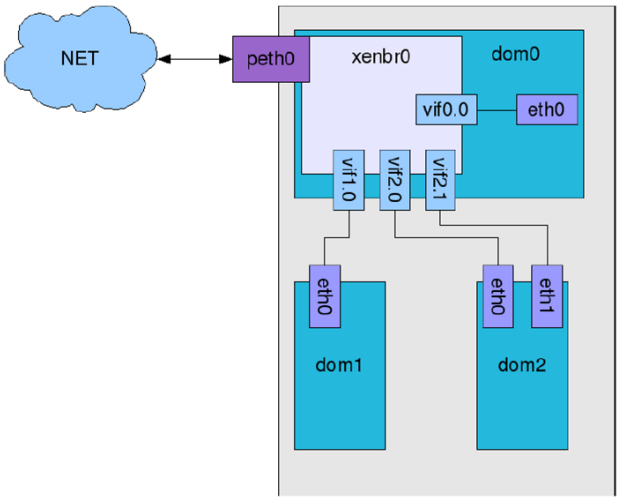

Figure 1. An example of how networking operates with XEN

A XEN framework consists of a first virtual machine, known as ***dom0*** or *privileged domain*, and a set of unprivileged machines named ***domU, U=1,...,N***; all ***domU*** virtual machines (each with its own OS, applications etc.) communicate with ***dom0*** to gain access to real hardware resources, being the ***dom0*** responsible for hardware and virtual resource management as well as for the creation, suspend, resume, and destroy of ***domU*** virtual machines.

Each time a new virtual machine is instantiated, it can have different (virtual) network interfaces, all of them associated to a corresponding set of virtual network interfaces in the dom0 thanks to a bridging mechanism, which ensures isolation of the network traffic in distinct virtual interfaces.

The association with ***dom0*** virtual interfaces is needed since physical networking is only managed by ***dom0***; XEN in particular makes the physical device accessible via a software-based switch, whose (virtual) ports are ***dom0*** virtual interfaces. An example to illustrate this mechanism is shown in Figure 1.

In this example, the physical network card managed by ***dom0*** is ***peth0***; this is accessible via the ***xenbr0*** bridge, the software-based switch whose virtual ports are ***vif1.0***, connected to the virtual interface ***eth0*** on **dom1**, and **vif2.0**, **vif2.1**, that are respectively the interfaces **eth0** and **eth1** of the virtual machine **dom2**. This approach allows any network configuration, as any **domU** can have its own number of interfaces; moreover the implementation is faithfully thanks to the traffic isolation guaranteed by xenbr0 for all virtual network interfaces. The simple and powerful configuration we described is leveraged to allow the creation of virtual environments, as discussed in the next section.

IV. THE VIRTUALNET TOOL

In this section we present ***VirtualNet*** a tool for virtualization-based network security assessment. In particular, as introduced previously, we leverage XEN and Qemu to instantiate and configure virtual hosts as well as to connect their interfaces to set up a duplicate of a real scenario, so that security test can be conducted in a faithfully, costs saving and risk-free fashion.

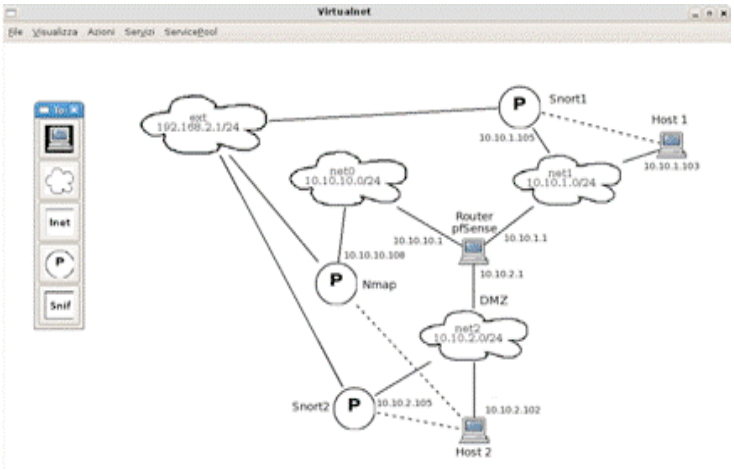

Figure 2. A Snapshot of VirtualNet

Other works based on virtualization exists, e.g. in [33] VMware is used to set up a virtual laboratory for network security; our proposal differs in the use of open source softwares (XEN and Qemu) and in the fact that our purpose is not to create a safe and secure environment (as it is deliverable for students in a virtual laboratory), rather ***VirtualNet*** can be used also to test disruptive scenarios, e.g. where DoS or penetration exploits attacks to server machines can be conducted to effectively test the duplicate of real environment.

A network consists of several elements, i.e. a set of hosts connected together, each one coming with a set of installed applications and services, and an infrastructure network they are connected to; such network can include devices as firewall, switch, routers and so on. In our approach is possible to create as many hosts as needed, each one with several virtual network interfaces, in order to connect them to any number of networks. Note that ***VirtualNet*** does not impose specific network topologies, however it allows us to insert virtual network devices (e.g. routers, dhcp servers) as needed; other works exploit virtualization just for such devices, e.g. [17] focuses on the development of a XEN-based firewall with *netfilter/iptables* software. Finally, note that ***VirtualNet*** allows us the creation of networks where virtual hosts are connected each other, in the sense that virtualization is used for hosts creation and configuration rather than for deploying a virtual network infrastructure. Other works (e.g., [8], [9], [31], [36]) focus on virtual networks, for instance how specific traffic patterns can be generated and are transmitted over the network, how virtual networks are mapped onto physical ones (overlay networks) and so on; here we do not address such issues.

VirtualNet has been written in Python language and it uses an XML-based format to save and load created scenarios. It also provides a simple and intuitive GUI through which all actions can be performed.

A snapshot of ***VirtualNet*** is given in Figure 2, where a scenario with a couple of hosts, some networks, a router and some probes to scan the networks is shown. To create hosts, the PC-like icon on the top of the floating toolbar of Figure 2 is used. Once the host is given a name, a corresponding icon will represent it; by right-clicking on that icon several options are available, in particular:

- the host network interface can be configured by providing standard TCP/IP parameters as IP address, netmask, gateway etc. ;
- the type of host is established; in particular the Qemu has been used in addition to XEN since it allows different OSes than classical Linux flavors (as for instance, Windows). This way up to four host virtualization types can be selected, i.e. XEN paravirtualization, XEN full virtualization, Qemu-based Windows or FreeBSD;
- the host can be started (i.e. virtual machine will be instantiated and run) and its services can be configured.

To define hosts services, the *ServicePool* item in the menu is used (Figure 3).

As shown in Figure, the new service is given a name (*service1* in figure), and the actual type of service can be selected from a list; basic network services are available from the list, in particular:

- the DNS, using the *Bind9* software;
- Firewalling (via *iptables* or *pfsense*);
- WWW service, using the *Apache* web server;
- Mail server, using the *Postfix* open source mailer;
- Routing, that allows TCP/IP packets forwarding;
- DCHP service, via the *dhcpd* server daemon;
- Switch, that allows the host to act as a network switch.

The GUI also provides all standard configurations for listed services.

Figure 3. The creation of a service

In addition to hosts, which are the main elements, other components can be created using the toolbar in Figure 2, in particular the cloud-shaped icon allows to create a network which is given an identifier and a range of addresses in the form of 32-bit mask. A particular network can be created with the *Inet* icon on the toolbar; it represents the ***xenbr0*** switch on ***dom0***, i.e. the physical network connected to the outside world.

The last elements ***VirtualNet*** allows us to add concern the network security, indeed they are the *probes* (denoted with the ***'P'*** icon in the toolbar shown in Figure 2) and the *sniffers*, instantiable via the ***'Snif'*** icon.

The *probe* is a host that comes with a port scanner and with an intrusion detection system (in our case we chose respectively Nmap and Snort), whose combination make probe the right object for network monitoring. The *probe* can be configured as desired, being a XEN-based virtual machine like any other host, in particular it is provided with the IP address of the *target machine*; i.e. the host acting placed in a strategic position in the virtual network in order to effectively analyze the network.

The *sniffer* is a host that exploits WireShark [22] to sniff packets and logs the traffic into a file for further analysis; similarly as for the *probe*, the *sniffer* must be provided with the IP address of the host to monitor. All hosts, included the special hosts *probe* and *sniffer*, will be connected together through networks, which are used as interconnecting elements.

VirtualNet allows to save a virtual scenario in a XML-based format to provide human-readability and ease of information exchanging; an example is given in Figure 4.

After the scenario has been created with the desired number and type of hosts and networks, in order to reproduce a real environment, the assessment of security should be performed. This can be accomplished manually by connecting to *probe* and/or *sniffer* objects; however ***VirtualNet*** also offers the possibility that a *probe* performs a network validation. In particular, right–clicking on a previously created *probe* a menu appears (see Figure 5), where a given host (or a whole network) can be specified as a target, together with a range of ports.

Then, a test can be selected from a list, e.g. the TCP SynPing, the TCP Ack Ping or others (a brief description is given in the menu); in addition, the probe can be set to discover both the OS and the services available at target hosts. When the *probe* will be started, it will execute the selected test and produce an output XML file; such results can also be shown on the GUI or saved for further processing.

Finally, ***VirtualNet*** can also perform all tests in an automatic fashion, in particular first via a host discovery then executing tests listed in Figure 5 sequentially, producing an XML file for each test. This mode can be used as a first network validation; if the security of the virtual environment is not acceptable, more specific, manual tests can be further executed.

V. ATTACK SCENARIOS

After having introduced ***VirtualNet***, here we illustrate some realistic scenarios where security assessment is performed. In particular, we conducted a penetration test, seeking for services and hosts vulnerabilities in the scenario of Figure 6.

```
<?xml version=1.0"?>
<structure>
    <Host id = "Host0">
        <iface id = "eth0">
            <ip>192.168.2.1></ip>
            <netmask> 255.255.255.0 </netmask>
            <gateway> 192.168.2.1 </gateway>
            <broadcast> 192.168.2.2 </broadcast>
                        <inet>net0</inet>
        </iface>
        <type id="Linux"/>
    </Host>
        <network addr="192.168.3.10/24"
     id="netext" linktoexternal="yes"/>
    <network addr="192.168.3.0/24"
     id="netext" linktoexternal="no"/>
</structure>
```

Figure 4. An example of XML-based file managed by VirtualNet

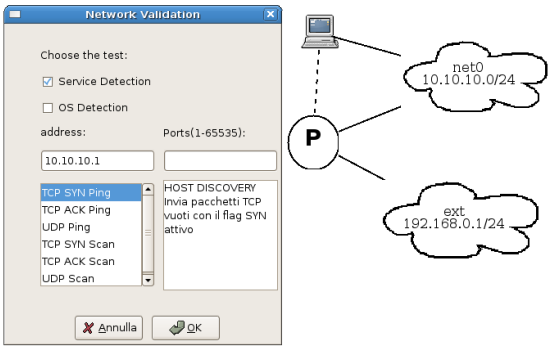

Figure 5. A probe network validation menu

In this environment, two networks are present, i.e. ***net0*** with IP range ***10.10.10.0/24*** and ***net1*** with IP range ***10.10.0.0/24***; the traffic between them is forwarded via the Router indicated in the figure, and whose IP addresses on both networks are respectively the ***10.10.10.1*** and the ***10.10.0.1***. In the ***10.10.0.0/24*** network, we also placed two hosts, the ***Host 0*** with IP the ***10.10.0.3*** and ***Host 1*** with IP the ***10.10.0.2***.

This scenario models the frequent situation in which some hosts are connected to an intranet (here, the ***10.10.0.0/24***) to be protected from external attacks, and a router acts as a firewall to properly filter the traffic coming from the outside (here, the ***10.10.10.0/24*** network); to this purpose, the *Router* was a XEN powered virtual machine with a Debian Linux, with enabled packet forwarding and *netfilter/iptables* as a firewall software.

To complete the scenario, three probes have been created, the ***Snort 1*** with ***IP 10.10.10.8*** used to analyze the traffic over the ***10.10.10.0/24*** network, the ***Nmap*** with IP ***10.10.10.4***, used to perform attacks to the victim host ***Host 1***, and the ***Snort 2*** with IP ***10.10.0.8*** used to analyze the traffic over the ***10.10.0.0/24*** network, in particular to detect the attacks from ***Nmap***. The dashed lines connecting ***Nmap*** and ***Snort 2*** with ***Host 1*** denotes fact that ***Host 1*** is the victim. Finally, they interface to the ***192.168.2.0/24*** network, which is a control network (being not part of the scenario) used to provide SSH access to all probes for management purposes, i.e. to launch the penetration test.

Test was started by selecting the automatic *Network validation* for the ***Nmap*** probe, as introduced previously (see Figure 8, where the target IP was set to ***10.10.0.2***).

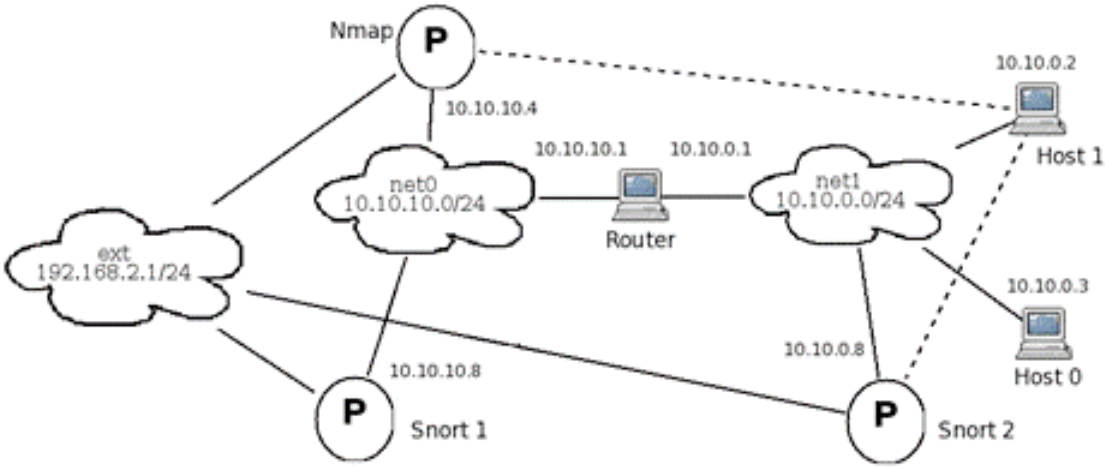

Figure 6. The first scenario used for security assessment test

During the first test, the firewall did not block any traffic, i.e. ***net1*** was not protected; after the test has been performed, the output provided by ***Snort 2*** is shown in Figure 7, where the probe revealed successful attacks coming from ***Nmap*** (***ALERTS:3*** in figure).

In a second test, firewall rules were enforced to protect the net1, including the target ***Host 1***; this time, the same attack coming from ***Nmap*** was blocked, indeed the probe ***Snort 1*** detect the attacks since they come from the same network of ***Nmap***, i.e. ***net0***, but ***Snort 2*** does not report any suspect traffic, thanks to the protection provided by the firewall at the *Router* (outputs are not shown). Similar test were conducted replacing the *Router* with a virtual machine powered by Qemu with FreeBSD as OS and *pfSense* as firewall, getting similar results, i.e. successful attacks when *pfsense* does not filter any traffic and an acceptable protection for ***Host 1*** when the firewall is properly configured.

A second scenario for security assessment is represented in Figure 8, where ***net1*** represents a *Militarized Zone* (MZ), i.e. an internal network to protect that includes ***Host 1*** and the monitor probe ***Snort1***. The network ***net2*** represents a *DeMilitarized Zone* (DMZ), i.e. a network where vulnerable services (e.g., www) will be placed; inside this network, ***Host 2*** is the victim of attacks, monitored by the probe ***Snort2***. The network ***net0*** represents the outsideworld (i.e. the internet), where attacks come from; in particular the attacker is the probe *Nmap*. The *Router* connects the networks and also acts as a firewall (*pfSense*). Finally, the network ext is used to manage all probes via SSH (as for the first scenario).

```
Snort received 3197 packets
    Analyzed: 3197(100.000%)
    Dropped: 0(0.000%)
===============================================================================
Breakdown by protocol:
    TCP: 3187       (99.687%)
    UDP: 0          (0.000%)
   ICMP: 0          (0.000%)
    ARP: 10         (0.313%)
  EAPOL: 0          (0.000%)
   IPv6: 0          (0.000%)
    IPX: 0          (0.000%)
  OTHER: 0          (0.000%)
DISCARD: 0          (0.000%)
===============================================================================
Action Stats:
ALERTS: 3
LOGGED: 4
PASSED: 0
===============================================================================
TCP Stream Reassembly Stats:
    TCP Packets Used: 1483       (46.387%)
    Stream Trackers: 786
    Stream flushes: 0
    Segments used: 0
    Stream4 Memory Faults: 0
===============================================================================
Final Flow Statistics
,----[ FLOWCACHE STATS ]----------
Memcap: 10485760 Overhead Bytes 16400 used(%1.701307)/blocks (178395/906) Overhead blocks
IPV4 count: 905 frees: 0 low_time: 1251798961, high_time: 1251798962, diff: 0h:00:01s
    finds: 1483 reversed: 578(%38.975051)
    find_sucess: 578 find_fail: 905 percent_success: (%38.975051) new_flows: 905
 Protocol: 6 (%100.000000) finds: 1483  reversed: 578(%38.975051)
  find_sucess: 578 find_fail: 905 percent_success: (%38.975051) new_flows: 905
device eth1 left promiscuous mode
audit(1251798967.494:8): dev=eth1 prom=0 old_prom=256 auid=4294967295
Snort exiting
```

Figure 7. Snort ouput in the first security assessment scenario

In a first experiment, *Nmap* was used to attack both the MZ and the DMZ, with the firewall disabled; the output of *Nmap* is shown in Figure 9, where the attacker discovered both hosts, i.e. ***Host 1*** on ***net1*** and ***Host 2*** on ***net2***.

Further, the firewall was properly configured to filter attacks, in particular the traffic coming from net0 is blocked (deny policy) when directed to the MZ network, and allowed if directed to the DMZ network, whereas DMZ and MZ can communicate each other, as normally occurs in a real scenario. Starting the test with the probe *Nmap*, this time ***Snort 2*** detects attacks coming from the outside and directed to the DMZ (29 alerts in Figure 9), whereas the ***Snort 1*** does not reveal any attack, showing that MZ is effectively protected by the firewall.

In real environments, traffic to the DMZ is generally not totally allowed, rather proper filtering policy are established at the firewall, so that only non-malicious traffic is allowed; here we just aimed at performing some tests to validate ***VirtualNet*** in security assessment. The tool has been revealed useful since all tests were performed in a risk-free fashion, i.e. the victim of attacks were virtual hosts rather than real servers, so any experiment can be conducted safely, yet preserving its accuracy in reproducing real conditions.

VI. CONCLUSIONS

In this work the question of network security has been considered, introducing the ***VirtualNet*** tool that exploits virtualization to set up a virtual environments (virtual hosts and networks), where security assessment can be safely and faithfully performed on duplicate of real scenarios. We presented ***VirtualNet*** and its capabilities, also conducting penetration tests in two real scenarios, showing the usefulness of our proposal.

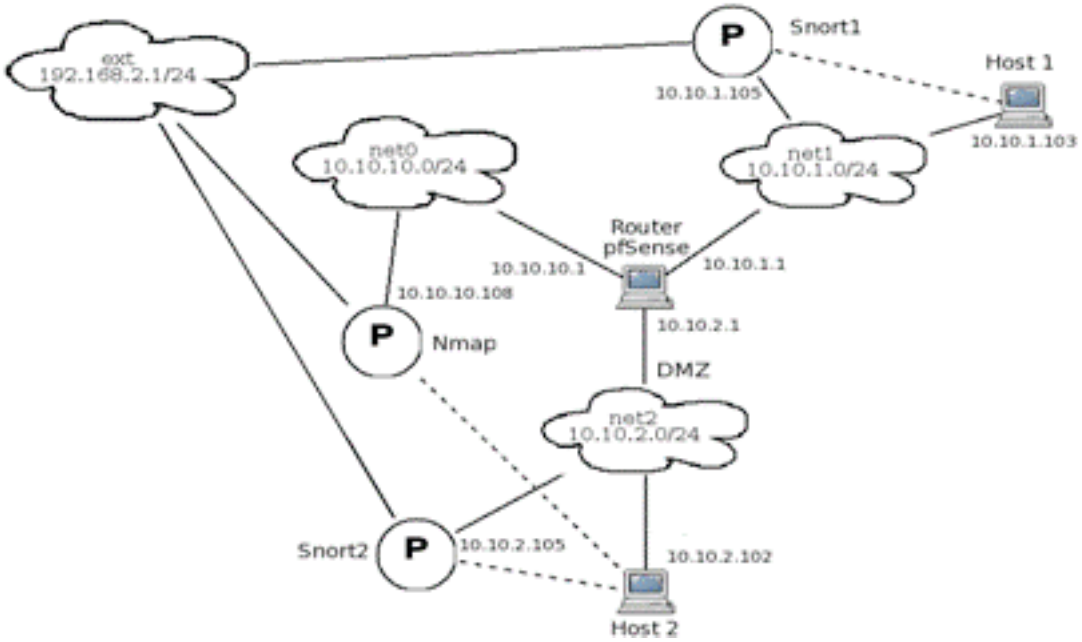

Figure 8. The second scenario used for security assessment test

VII. FUTURE WORKS

This work deserves some considerations about future ideas, in particular:

first, security assessment in more complex scenarios should be performed, in order to completely validate ***VirtualNet***; several attacks could be tested, as for instance DNS poisoning, SymLink attacks, SYN flooding, smurf attacks, DoS, DDos and others;

the virtual network creates ***VirtualNet*** does not take into account bandwidth, QoS and other features; to allow experiments in different network conditions, this issue should be addressed (as for instance in [8]) ;

VirtualNet performs security tests automatically but the position of probes is manually selected; a further evolution of the tool concerns semi-automatic functionality that helps to discover where attacks can come from, and which countermeasures could be adopted (e.g. how firewall should be configured).

```
Snort received 13472 packets
    Analyzed: 13472(100.000%)
    Dropped: 0(0.000%)
===============================================================================
Breakdown by protocol:
    TCP: 13445      (99.800%)
    UDP: 6          (0.045%)
   ICMP: 14         (0.104%)
    ARP: 6          (0.045%)
  EAPOL: 0          (0.000%)
   IPv6: 1          (0.007%)
    IPX: 0          (0.000%)
  OTHER: 0          (0.000%)
DISCARD: 0          (0.000%)
===============================================================================
Action Stats:
ALERTS: 29
LOGGED: 32
PASSED: 0
===============================================================================
TCP Stream Reassembly Stats:
    TCP Packets Used: 6288      (46.675%)
    Stream Trackers: 3163
    Stream flushes: 0
    Segments used: 0
    Stream4 Memory Faults: 0
===============================================================================
Final Flow Statistics
,----[ FLOWCACHE STATS ]----------
Memcap: 10485760 Overhead Bytes 16400 used(%5.840969)/blocks (612470/3331) Overhead blocks: 1 Could Hold: (58579)
IPV4 count: 3330 frees: 0 low_time: 1251797341, high_time: 1251797368, diff: 0h:00:27s
    finds: 6308 reversed: 2946(%46.702600)
    find_sucess: 2978 find_fail: 3330 percent_success: (%47.209892) new_flows: 3330
 Protocol: 1 (%0.221940) finds: 14  reversed: 4(%28.571429)
  find_sucess: 11 find_fail: 3 percent_success: (%78.571429) new_flows: 3
 Protocol: 6 (%99.682942) finds: 6288  reversed: 2942(%46.787532)
  find_sucess: 2963 find_fail: 3325 percent_success: (%47.121501) new_flows: 3325
 Protocol: 17 (%0.095117) finds: 6  reversed: 0(%0.000000)
  find sucess: 4 find fail: 2 percent success: (%66.666667) new flows: 2
```

Figure 9. Snort results for DMZ in the second scenario

REFERENCES

[1] K. Adams and O. Agesen, “A comparison of software and hardware techniques for x86 virtualization”, in Proceedings of the 12th international conference on Architectural support for programming languages and operating systems, ser. ASPLOS-XII. New York, NY, USA: ACM, 2006, pp. 2–13.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, “Xen and the art of virtualization”, SIGOPS Oper. Syst. Rev., vol. 37, pp. 164–177, October 2003. http://dx.doi.org/10.1145/1165389.945462

[3] G. Bell and P. Dourish, “Yesterday’s tomorrows: notes on ubiquitous computing’s dominant vision”, Personal Ubiquitous Comput., vol. 11, no. 2, pp. p. 133–143, 2007.

[4] C. M. Buechler, J. Pingle, Michael W. Lucas, "pfSense: The Definitive Guide" Reed Media Services; 1st edition (November 1, 2009) ISBN-10: 0979034280

[5] F. Bellard, “Qemu, a fast and portable dynamic translator”, in Proceedings of the annual conference on USENIX Annual Technical Conference, ser. ATEC ’05. Berkeley, CA, USA: USENIX Association, 2005, pp. 41–41.

[6] A. Bürkle, A. Hertel, W. Müller, and M. Wieser, “Evaluating the security of mobile agent platforms”, Autonomous Agents and Multi-Agent Systems, vol. 18, pp. 295–311, April 2009. http://dx.doi.org/10.1007/s10458-008-9043-z

[7] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, “Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility”, Future Generation Computer Systems, vol. 25, no. 6, pp. 599 – 616, 2009. [Online]. http://dx.doi.org/10.1016/j.future.2008.12.001

[8] J. Carapinha and J. Jimenez, “Network virtualization: a view from the bottom”, in Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures, ser. VISA ’09. New York, NY, USA: ACM, 2009, pp. 73–80.

[9] N. M. K. Chowdhury and R. Boutaba, “A survey of network virtualization”, Comput. Netw., vol. 54, pp. 862–876, April 2010. http://dx.doi.org/10.1016/j.comnet.2009.10.017

[10] P. Dadhich, D. Dutta, and D. M.C.Govil, “Security issues in mobile agents”, International Journal of Computer Applications, vol. 11, no. 4, pp. 1–7, December 2010, published By Foundation of Computer Science.

[11] I. T. Foster, “The anatomy of the grid: Enabling scalable virtual organizations”, in Proceedings of the 7th International Euro-Par Conference Manchester on Parallel Processing, ser. Euro-Par ’01. London, UK: Springer-Verlag, 2001, pp. 1–4. [Online].

[12] C. Garrigues, S. Robles, J. Borrell, and G. Navarro-Arribas, "Promoting the development of secure mobile agent applications", J. Syst. Softw., vol. 83, pp. 959–971, June 2010. http://dx.doi.org/10.1016/j.jss.2009.11.001

[13] R. P. Goldberg, "Architecture of virtual machines", in Proceedings of the June 4-8, 1973, national computer conference and exposition, ser. AFIPS '73. New York, NY, USA: ACM, 1973, pp. 309–318.

[14] B. Hay, "Applications of virtualization to digital forensics education", in Proceedings of the 2010 43rd Hawaii International Conference on System Sciences, ser. HICSS '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 1–7.

[15] C. Kaufman, R. Perlman, and M. Speciner, "Network security: private communication in a public world", second edition, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2002.

[16] D. Kotz and R. S. Gray, "Mobile agents and the future of the internet", SIGOPS Oper. Syst. Rev., vol. 33, pp. 7–13, July 1999. http://dx.doi.org/10.1145/311124.311130

[17] F. Liu, X. Su, W. Liu, and M. Shi, "The design and application of xen-based host system firewall and its extension", in Proceedings of the 2009 International Conference on Electronic Computer Technology. Washington, DC, USA: IEEE Computer Society, 2009, pp. 392–395. http://dx.doi.org/10.1109/ICECT.2009.83

[18] G. F. Lyon, Nmap Network Scanning, "The Official Nmap Project Guide to Network Discovery and Security Scanning". USA: Insecure, 2009.

[19] S. Nanda and T. cker Chiueh, "A survey of virtualization technologies - technical report", Department of Computer Science, SUNY at Stony Brook, Tech. Rep., 2005.

[20] Netfilter/Iptables project. Available:http://www.netfilter.org

[21] E. Nieuwdorp, "The pervasive discourse: an analysis", Comput. Entertain., vol. 5, no. 2, p. p. 13, 2007. http://dx.doi.org/10.1145/1279540.1279553

[22] A. Orebaugh, G. Ramirez, and J. Burke, "Wireshark and Ethereal network protocol analyzer toolkit", ser. Jay Beale's open source security series. Syngress, 2007.

[23] S. Paul, J. Pan, and R. Jain, "Architectures for the future networks and the next generation internet: A survey", Comput. Commun., vol. 34, pp. 2–42, January 2011. http://dx.doi.org/10.1016/j.comcom.2010.08.001

[24] A. Ribiere, "Using virtualization to improve durability and portability of industrial applications", in Proceedings of the 6th IEEE International Conference on Industrial Informatics, INDIN 2008. IEEE Press, 2008, pp. 1545–1550.

[25] M. Roesch, "Snort - lightweight intrusion detection for networks", in Proceedings of the LISA 1999, 13th Systems Administration Conference. Usenix association, 1999.

[26] M. Roesch et alii, "Snort Users Manual". Available: http://www.snort.org

[27] A. V. Romero, "VirtualBox 3.1: Beginner's Guide" Packt 2010

[28] J. Sahoo, S. Mohapatra, and R. Lath, "Virtualization: A survey on concepts, taxonomy and associated security issues", International Conference on Computer and Network Technology, pp. 222–226, 2010.

[29] W. Stallings, "Network security essentials", 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2007.

[30] J. Sugerman, G. Venkitachalam, and B.-H. Lim, "Virtualizing i/o devices on vmware workstation's hosted virtual machine monitor", in Proceedings of the General Track: 2002 USENIX Annual Technical Conference. Berkeley, CA, USA: USENIX Association, 2001, pp. 1–14.

[31] J. Touch, Y. Wang, L. Eggert, and G. Finn, "A virtual internet architecture", in Proceedings of the ACM SIGCOMM Workshop on Future Directions in Network Architecture. New York, NY, USA: ACM, 2003.

[32] T. Thein, M. Pokharel, S.-D. Chi, and J. S. Park, "A recovery model for survivable distributed systems through the use of virtualization", in Proceedings of the 2008 Fourth International Conference on Networked Computing and Advanced Information Management - Volume 01, ser. NCM '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 79–84. http://dx.doi.org/10.1109/NCM.2008.213

[33] X. Wang, G. C. Hembroff, and R. Yedica, "Using vmware vcenter lab manager in undergraduate education for system administration and network security", in Proceedings of the 2010 ACM conference on Information technology education, ser. SIGITE '10. New York, NY, USA: ACM, 2010, pp. 43–52.

[34] R. Want and T. Pering, "System challenges for ubiquitous & pervasive computing", in ICSE '05: Proceedings of the 27th international conference on Software engineering. New York, NY, USA: ACM, 2005, pp. p. 9–14.

[35] C. Weltzin and S. Delgado, "Using virtualization to reduce the cost of test", in Proceedings of the AUTOTESTCON, 2009 IEEE. IEEE Press, 2009, pp. 439–442.

[36] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components", in Proceedings of the INFOCOM 2006. 25th IEEE International Conference on Computer Communications. IEEE, 2006.

AUTHORS

Vincenza Carchiolo is currently full professor of Computer Science at DIEEI - University of Catania. Her research interests include information retrieval, query languages, distributed system, and formal language. She received a degree with Honors in Electrical Engineering from University of Catania in 1983 (Vincenza.Carchiolo@dieei.unict.it).

Alessandro Longheu received his MS in Computer Engineering in 1997 from the University of Catania and then his PhD in 2001 from the University of Palermo. He taught Programming Languages, Computer Networks, Data Management and Advanced programming at the University of Catania and UniKore of Enna. His research interests include e-learning, workflows, information retrieval and integration in the semantic web, complex networks and trust and semantic web (Alessandro.Longheu@dieei.unict.it).

Giuseppe Mangioni is assistant professor at DIEEI - University of Catania. He received the Ph.D. degree in 2000 at the University of Catania. Currently he is professor of Computer Networks at the Faculty of Engineering of Catania. His research interests include peer-to-peer systems, trust/reputation systems, self-organizing and self-adaptive systems and complex networks (Giuseppe.Mangioni@dieei.unict.it).

Michele Malgeri is associate professor in Department of informatics and Telecommunications at University of Catania. His research interests include distributed system, information retrieval, query languages and formal language. He received a degree with Honors in Electrical Engineering from University of Catania, Italy in 1983 (Michele.Malger@dieei.unict.it).

Submitted 09 August 2013. Published as resubmitted by the authors 13 September 2014.