Cloud-based Testbed for Simulation of Cyber Attacks

Daniel Kouřil, Tomáš Rebok, Tomáš Jirsík, Jakub Čegan, Martin Drašar, Martin Vizváry, Jan Vykopal
Masaryk University, Institute of Computer Science
Botanická 68a, 602 00 Brno, Czech Republic
{*lastname*}@ics.muni.cz

***Abstract*—Cyber attacks have become ubiquitous and in order to face current threats it is important to understand them. Studying attacks in a real environment however, is not viable and therefore it is necessary to find other methods how to examine the nature of attacks. Gaining detailed knowledge about them facilitates designing of new detection methods as well as understanding their impact. In this paper we present a testbed framework to simulate attacks that enables to study a wide range of security scenarios. The framework provides a notion of real-world arrangements, yet it retains full control over all the activities performed within the simulated infrastructures. Utilizing the sandbox environment, it is possible to simulate various security attacks and evaluate their impacts on real infrastructures. The design of the framework benefits from IaaS clouds. Therefore its deployment does not require dedicated facilities and the testbed can be deployed over miscellaneous contemporary clouds. The viability of the testbed has been verified by a simulation of particular DDoS attack.**

I. Introduction

In order to be able to face the threats posed by contemporary attackers, it is crucial to continually develop techniques and methods for detection and prevention of attacks. Also, the nature of cyber attacks is evolving, so is the focus of attackers. This volatile character of computer attacks makes it challenging to catch up with current trends in the area. The situation is also complicated by the closed nature of the domain, where attackers naturally do not reveal details about their techniques, which often need to be found out only during real security incidents. In order to efficiently confront contemporary attacks it is inevitable to understand their nature and to develop mechanisms for their detection.

A viable option to study attacks is their simulation. Simulating an attack in a controlled environment makes it possible to understand the impact of the attack on individual services in the infrastructure, the way how the attack is spreading, what is the role of all peers, etc. Such deep comprehension of attacks then significantly contributes to design and operation of services that are more resilient to contemporary attacks and threats. For instance, during the last two years we have been witnessing an increase of various types of distributed denial-of-service (DDoS) attacks ([1], [2]). For deployed services it is important to estimate the size of the attack they are able to withstand, which can be verified by simulation of DDoS attacks of different volumes.

Besides being able to design proper protection, one also needs to recognize attacks reliably and soon enough. However, crafting proper detection tools and methods is a tedious process, especially when it comes to decreasing the number of false positive alarms. Development of the tools and their verification can be significantly streamlined with a suitable testbed where various attack scenarios can be pursued. In order for such a testing environment to be helpful, it needs to be able to simulate real-world attacks and enable studying them.

978-1-4799-0913-1/14/$31.00 © 2014 IEEE

The need for an environment where attacks can be simulated and studied poses questions about feasibility of simulation of cyber attacks. First, it is necessary to answer the question whether it is even possible to build an artificial environment that resembles as much as possible the real world, yet it provides sufficient isolation and control over all activities performed within it. Second, can we find the right balance between flexibility and usability of the environment, i.e. can we hide unnecessary configuration details without losing the possibility to support any configuration of networks and hosts? And, third, given such an environment exists, is it possible to describe an arbitrary attack and model it in the environment so it can be studied properly?

In this paper, we propose a novel framework that we have developed to establish a testbed for simulation of various cyber attacks. The framework is designed for, and built upon an existing cloud infrastructure where different virtual appliances can be easily deployed, which provides a flexible layer for additional configurations. In-depth monitoring is an inherent part of the testbed. It allows testbed users to transparently observe network communication among individual network nodes as well as to obtain details about nodes performance. By utilizing the testbed it is possible to execute various security experiments to simulate cyber attacks and study them.

II. Cloud-based security testbed

There are many testbed solutions intended to support security-related simulations in various manners. Some of them, namely DETER [3] and TWISC [4], employ the generic and publicly available *Emulab/Netbed* [5] infrastructure solution, which provides them with basic functionality for virtual appliances' deployment, flexible network topologies configuration, various network characteristics emulation, etc. While Emulab simplifies many tasks related to deploying and building various network topologies for the experiments, it brings several restrictions on both the infrastructure as well as its features - these include simulated topology restrictions (e.g., just IPv4 is supported) as well as restrictions related to building the infrastructure itself (e.g., several OS and HW restrictions have to be met).

In contrast, several security-related testbeds require their own infrastructure solution to be established, which cannot be used for other purposes. For example, ViSe [6], LVC [7], and V-NetLab [8] testbeds employ the VMware virtualization, while the hypervisor-based security testbed [9] requires a KVM-based infrastructure. All these cases require to purchase and establish a dedicated infrastructure, which brings both strengths and weaknesses by itself—while the full control over the infrastructure can lead to easier deployment of testbed's features, it also leads to high initial costs and limited growth-flexibility.

A. Requirements for the intended testbed

Considering the purpose and foreseen use of the intended testbed infrastructure we have identified a set of requirements that the testbed must fulfill to meet our needs. For the sake of clarity, we divide these requirements into five categories—network-related, hosts-related, monitoring, testbed control, and deployment requirements.

Regarding the *network-related requirements*, the testbed's ability to define and run any network topology is obviously necessary, whether it is a single node or several interconnected networks. The testbed has to allow users to have complete control over the network Layer 3 arrangement. This feature, which is not available in most of the existing security-related testbed solutions that we studied, is required to use an arbitrary L3 protocol and/or an addressing schema, including public IP addresses (within a sandboxed environment).

In order to approach real-world arrangements, additional network characteristics should be supported. To simulate various networking types, like ADSL modems or mobile devices, the testbed should provide ability to emulate various network properties, such as limited bandwidth, delays, packet drop rate, or link failures. While we foresee an isolated environment is used for most simulations, the testbed must also support scenarios which require connections to real Internet servers. An example of such a scenario is an examination of the protocol between an infected computer and a real malicious server. Such traffic must be filtered properly using firewall rules.

The *hosts-related requirements* concern the possibilities to support various hosts configurations. To provide sufficient flexibility, the testbed has to be able to establish nodes running common operating systems and architectures.

To provide *monitoring features*, the testbed is required to monitor network links between any two nodes in the defined virtual topology and collect monitoring data about network flows or even packet dumps of the entire communication passed over the emulated wires. Besides network-based probes, host-based probes (e.g., CPU and memory usage monitoring) should also be provided. Naturally, all the monitoring functionality has to be performed transparently, so it cannot be detected or even results could not be distorted.

The testbed *control requirements* basically include possibilities to orchestrate the testbed and to control all its components easily. The testbed has to expose a user interface that is general enough yet easy to use for users so that they do not need to cope with internal configuration details. The testbed should further give an option to perform a defined set of operations in real time, allowing for an interactive intervention to and/or control of running attacks. Similarly, attack scenarios themselves should also be easy to set-up, deploy and execute. The testbed should support repetitions of the same experiments multiple times, which is required, for instance, when tuning and evaluating a particular detection mechanism.

Concerning the *deployment requirements*, the testbed must expect just widely-used middleware for testbed operations so that one is able to deploy it over an existing cloud-based infrastructure providing supported interfaces.

Given the features we require from such a security-related testbed, capabilities of the existing solutions, as well as benefits of the cloud-based infrastructures, we decided to propose a new testbed employing cloud infrastructures providing an infrastructure as a service (IaaS clouds).When using well-known cloud interfaces, the testbed need not rely on a particular cloud provider but can be easily adapted to any of them, assuring reasonable costs and providing flexibility when necessary. Unlike to other solutions, we provide a general framework to build own testbeds, which can deployed in different environments. This approach has its limitation, with the main one being the configuration of networks. Solely using widely-used networking mechanisms, we introduce a novel approach to building flexible virtualized networks.

B. Proposed solution

The proposed framework is built on top of a cloud, which provides an abstract layer hiding the issues related to the maintenance of virtual appliances, and which allows the desired growth-flexibility. Every simulation is provided with a sandbox where attacks can be safely executed. The whole sandbox environment is instantiated in a cloud and comprises solely of virtual machines. In addition to the control of the cloud layer, the framework also provides means to define descriptions of security experiments and to monitor and control their executions. Given the resemblance with similar environments from other areas (such as warfare) we call the framework *Cybernetic Proving Ground* (*CPG* in short).

CPG allows users to configure multiple virtual environments for simultaneous run of multiple simulations. Every environment is provided with a dedicated service node (*Scenario Management Node*) that is used to operate the established virtual environment. The node also provides the entry point to the virtual environment and hosts auxiliary services like a collector for monitoring data, etc.

The nodes within the environment are connected by a virtual network. Building such a framework on top of a cloud providing limited networking flexibility (in terms of topologies, L3 addressing, L3 protocol, etc.) brought several challenges we had to cope with. Not to depend on a particular cloud implementation/infrastructure, we decided to avoid running any *CPG* service tools within the underlying cloud layer—thus, we had to ensure defined data paths (to allow flows monitoring within virtual appliances) as well as L3 layer flexibility, all of which using common networking mechanisms provided by common cloud implementations. To overcome these limitations and to satisfy the prescribed network-related requirements, we introduce a concept of so-called *LAN management nodes (LMNs)* providing an abstrac-

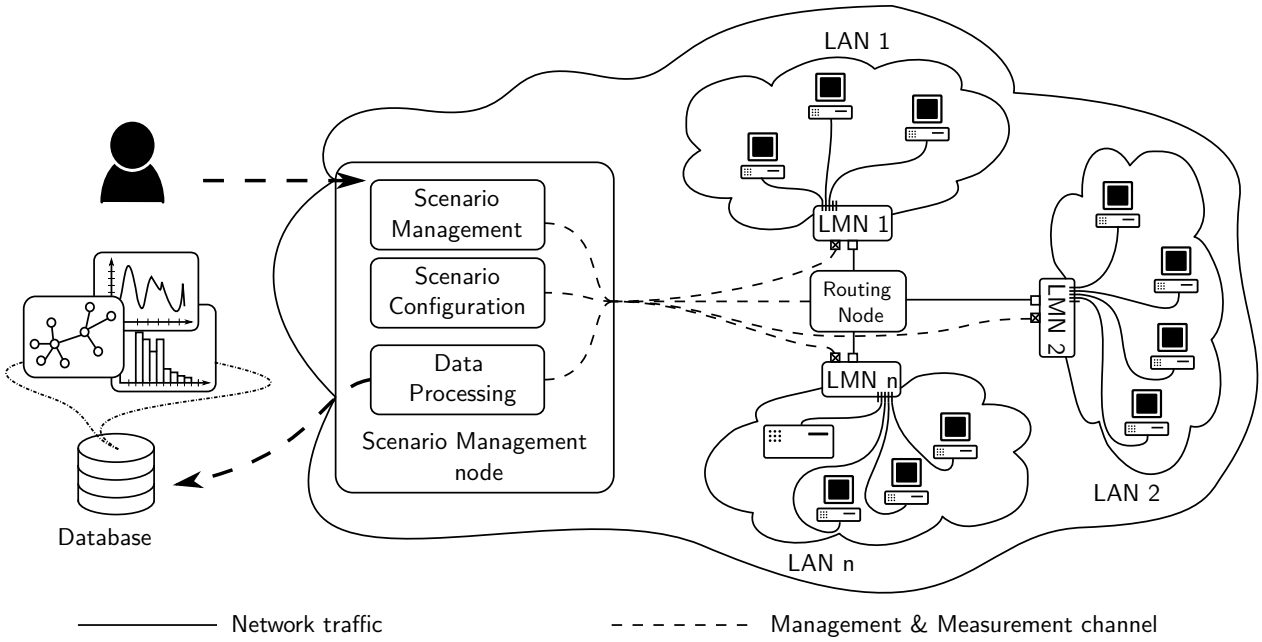

Fig. 1. A schema of a simulation environment

tion of a LAN within the *Cybernetic Proving Ground*. These nodes, running as common virtual machines within the cloud, play a role of virtual switches invisible to users—they run a common OS with an implementation of a virtual switch (Open vSwitch [10] in our current prototype) and are equipped with a number of network interfaces. Using standard cloud services, each network interface is then interconnected with a single node of the particular virtual LAN or with a routing node (see Fig. 1)—the interconnection is provided by common L2 services (VLANs), representing an illusion of a virtual cable established between every node of the user's LAN and a virtual switch. Every interconnection has to be equipped with a VLAN tag[1] that is unique throughout the *CPG* infrastructure, thus establishing a separate VLAN network and ensuring the necessary isolation among different virtual links. As all the data processing operates on L2, the employed L3 protocols as well as addressing schema employed for the internal LANs are thus kept in the hands of *CPG* users.

The usage of LMNs provides further benefits besides those already mentioned—since all the data sent within the internal user's LAN(s) has to traverse through the LMNs, these nodes also serve for transparent network monitoring, transparent emulation of networking properties (e.g., limited bandwidth, link failures, packet delay, packet loss) using standard Linux kernel's tools like netem and traffic control. As depicted in the Fig. 2, the LMNs may be further equipped with other useful services—for example, a DHCP server for the LAN, or a firewall filtering the data passing between different user's LANs or from/to external WAN.

To allow multiple virtual LANs within an environment as well as communication among them, the *Routing Node* is employed. Besides routing and optional filtering on the data network, the node also allows to monitor data flows passing among the LANs. Additionally to the data network, a *management network* is used for delivering control and measurement information among the LMNs, Routing Node, and the Scenario Management Node. For now, we decided not to equip the user virtual nodes with an interface to the control network, so that an attack trying to deploy itself through all the accessible interfaces cannot perform actions out of control. Rather, we employ common cloud remote-access interfaces to allow the user to access the virtual nodes' console.

To fulfill the requirements for the testbed monitoring, a very dense monitoring infrastructure is deployed. Since we need to monitor both network communication and nodes performance, we deploy a network monitoring infrastructure and host monitoring infrastructure transparently in each testbed created.

The network monitoring infrastructure is a set of probes, data processing unit and a database. The probe is a unit which runs the metering and exporting processes. It gathers information about traffic from the network. To enrich information acquired, we have developed various import plugins enabling application layer monitoring (e.g., HTTP [12], geolocation [13]). The probe is also capable of dumping the whole network traffic, not only the packet headers. The *Cybernetic Proving Ground* makes available various types of probes (e.g., FlowMon or nProbe) that can be selected for an experiment execution.

The probes are deployed at specific observation points. In order to reach high-level visibility, we make the observation points an integral part of the LAN management node of each LAN (see Fig. 2). The probes are connected to the network via the mirror port (SPAN) of the virtual switch. This kind of probe deployment gives the possibility to monitor traffic within the specific LAN as well as traffic among LANs. The acquired data is sent to the data processing unit located in

[1]To cope with the limited number of VLAN tags, the stacked VLANs (IEEE 802.1QinQ [11]) can be employed.

the Scenario Management Node via the management network. Several formats for the data representation are supported, namely NetFlow [14] and IPFIX [15].

The core of the data processing unit is a collector. The collector is a device dedicated for processing and storing data sent by probes. It further aggregates, preprocesses and stores the data. The *Cybernetic Proving Ground* supports multiple types of the collector, in particular NFDUMP and IPFIX-col. Additional scripts compute basic statistics and employ anomaly and attack detection methods (e. g., RdpMonitor [16], SSHCure [17]). The statistics and results of methods are sent to an external database for further processing and visualization.

The host monitoring infrastructure retrieves information about the node performance. It is capable of monitoring CPU and memory utilization, number of open connections, interface statistics and other host-based characteristics.

The *Cybernetic Proving Ground* exposes a command suite to establish and control the environment. The command suite has been designed to hide all the internal complexities of the environment so that the operator can focus solely on preparation of an attack simulation. For instance, the framework provides a small set of commands to establish a LAN inside the testbed, which encompasses provisioning of all auxiliary nodes, configuration of network links, setup of network probes, etc. The prototype implementation of the *Cybernetic Proving Ground* was developed for clouds managed by the OpenNebula [18] toolkit and the *CPG* framework is implemented using its command-line utilities.

The *Cybernetic Proving Ground* was designed to be as independent as possible on the underlying layer. Since the framework is implemented on top of a common cloud system, it can be easily moved to another IaaS cloud installation or even a standalone computer. In order for *CPG* to be deployed it is necessary that the cloud infrastructure gives sufficient control over the Layer 2, which is feasible with many cloud providers (especially in the research field).

A pilot of *CPG* was implemented in a cloud operated by Masaryk University and CESNET, the Czech National Research and Education Network. At the moment, the *Cybernetic Proving Ground* utilizes several hardware nodes of the infrastructure, which makes it possible to simulate environments composed of several LANs, each comprising a dozen of nodes. If a need for additional resources arises, the testbed can be easily enlarged, benefiting nicely from the cloud nature.

The *Cybernetic Proving Ground* enables users to create virtual environments that can be used for many activities like detailed forensics analysis of malware or security hands-on trainings. In the rest of paper, however, we describe how the environment can be used for attack simulations.

III. Modeling security scenarios

An experiment is completely described by a *scenario* and process of its realization. The *scenario* is defined as a group of nodes, logical and network topologies, monitoring rules, and a description of three phases of an attack simulation. Each of these parts has many available configuration parameters which determine the network environment. *Node* is a machine which is a member of the logical infrastructure and it is connected

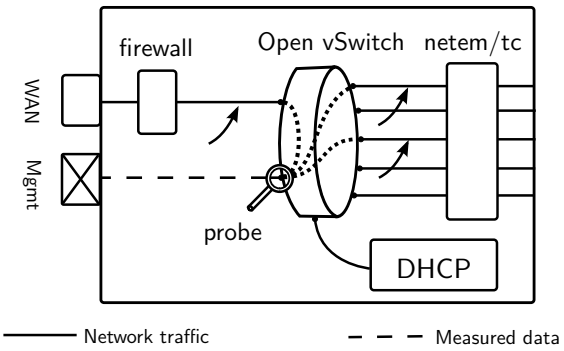

Fig. 2. Schema of LAN Management Node

to the network infrastructure via its configurable interface. It is possible to define node's hardware (CPUs and RAM), its operating system, and application software. *Network topology* describes interconnections between all nodes in a scenario. In order to simulate various networking arrangements, such as ADSL modems or mobile devices, this topology is also configurable in several parameters such as packet loss and bandwidth. *Logical topology* describes the role (reflector, attacker, victim, etc.) of each node and network in the scenario. Once the scenario is completely defined, the *Scenario Management node* can execute the first of three phases (initialization, run, evaluation) of the attack simulation.

In the first phase, the *initialization*, network and logical topologies of the environment are established and parameters of the attack are set. This initialization is controlled by scenario management which instantiates the testbed in the *Cybernetic Proving Ground*. The testbed provides the network topology defined in the scenario with all requested monitoring rules applied.

In the second phase, the *scenario run*, the actual experiment is done. The attack is executed according to the scenario. Both network and host monitoring infrastructures capture data such as NetFlow, IPFIX and host information from selected nodes and networks. The collector captures received data and pass it to the Scenario Management Node for storage and further analysis. All data and characteristics are continuously transferred from the Scenario Management node to a visualization infrastructure where they are displayed in accordance with requirements specified by the scenario. A further description of visualization is out of the scope of this paper.

The third phase, the *evaluation*, serves for an analysis of the experiment. Captured data is stored for later work, scenario modifications and its re-run. The experiment can also be replayed by the visualization management in different speeds. It is especially useful in case of a security training, where it is possible to use this functionality for detail debriefing of the exercise.

IV. Proof of concept

To verify our concept, we prepared a scenario of an application DDoS attack deployed in the infrastructure of the *Cybernetic Proving Ground*. According to the scenario, we deployed the testbed, simulated the attack and measured all

outcomes of the simulation. We chose to orchestrate a *Low & Slow* attack against a web server as DDoS attacks are ubiquitous nowadays. The attack depletes web server's resources by opening many connections and sending incomplete HTTP requests. We chose to run two experiments to show the *CPG* functionality. In the first experiment the web server is equipped with a module to mitigate the attack, while in the second experiment we targeted the web server without the module. This section describes in detail the realization of the scenario and is divided into three parts according to the scenario description in Section III.

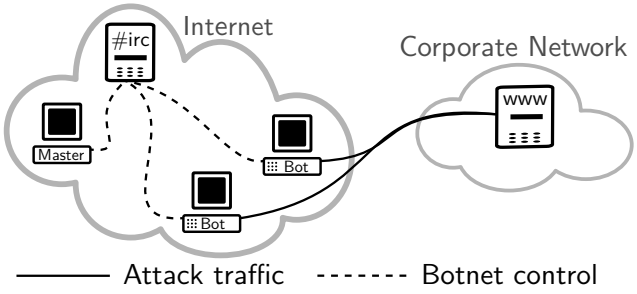

Fig. 3. Logical topology of the scenario of Low&Slow attack.

A. Initialization

The scenario is composed of four types of nodes, depicted in Fig. 3, to preserve authenticity of a real attack. The first type of a node is a victim. In our scenario, the victim is a Linux based operating system with the Apache web server. The second type is a bot. This node is the source of attack traffic against the victim. The third type is the attacker represented by the master node. This node gives commands to the botnet through the IRC server, which is the last node. The nodes were divided into two networks, one simulating the Internet with the attackers and another one hosting the victim web server (Corporate Network). The attack was performed with the *slowhttptest* [19] tool.

An instrumentation of the monitoring infrastructure is a part of the first phase. Considering the logical topology, the location of an observation point was selected on the edge of the corporate network. This allocation reflects typical deployment of the monitoring devices. Considering the monitoring facilities, we requested to use a flow-based probe FlowMon in the LAN Management Node on the ingress/egress interface. Thus we were able to monitor the incoming and outgoing traffic from/to the Corporate Network. Neither inside communication nor the communication between the master node and bots were monitored as it would only generate useless noise in this experiment. Moreover, we asked for a probe equipped with an additional input plugin retrieving information from HTTP headers [12]. An IPFIX collector was used to capture and store the data for further analysis.

B. Experiment Run

The run started by the IRC server that sent commands which made the bot launch the attack against the web server using the *slowhttptest* tool. The aim was to open 500 connections to the server in total at the rate of 20 connections per second and sent incomplete GET requests. Every 20 seconds another part of the requests was sent with the maximum size of 24 bytes. Moreover, every 5 seconds a probing GET request was sent to check the web server availability. The duration of both experiments was set to 600 seconds.

TABLE I. EXAMPLES OF CAPTURED FLOWS: # 1 - REGULAR TRAFFIC; # 2 - ATTACK AGAINST WEB SERVER WITH *enabled* MODULE FOR MITIGATION OF THE ATTACK; # 3 - ATTACK AGAINST WEB SERVER WITH *disabled* MODULE FOR MITIGATION OF THE ATTACK; REQ = REQUEST, RESP = RESPONSE

#	Type	Duration	Packets	Bytes	HTTP Code
1	REQ	0.002	7	612	-
	RESP	0.001	5	4203	200
2	REQ	75.050	12	959	-
	RESP	60.050	8	930	400
3	REQ	600.950	36	3034	-
	RESP	600.950	34	2282	400

In the first experiment, we attacked the web server with a *Low & Slow* attack mitigation module enabled. The module rejects all established connections that do not complete their request in the predefined timeout and returns HTTP error code 400 (Bad request). In the second experiment, we targeted the web server without the module enabled. The attack was successful in both cases, resources of the server were depleted and it became unavailable.

C. Experiment Evaluation

The third phase is the evaluation of the data captured by the monitoring infrastructure. Neither master, bots nor web server CPU or memory utilization measurement showed extraordinary peaks during the experiments. However, we observed a high number of established connections on the server side. This behavior corresponds to the nature of the *Low & Slow* attack. Examples of captured flows representing these connections are shown in Table I.

During the first experiment the server became unavailable after 14 seconds of the attack. However, as soon as the duration of the connection reached the timeout set by the mitigation module, the connection was terminated and the server returned HTTP code 400 . This experiment is represented by the flows # 2 in Table I. As the connections were terminated, the server become again available in 60 seconds. This time is consistent with the mitigation module timeout settings.

In the second experiment the server became unavailable after 14 seconds and remained in this state for next 586 seconds until the attack ended. As no mitigation module was activated, the *slowhttptest* tool kept all possible connections open and occupied. So the duration of the flows # 3 in Table I is equal to the experiment duration. We can observe that *slowhttptest* sent 32 attacking packets which is consistent with the experiment settings. The remaining 4 packets were used for connection establishment and termination.

D. Summary

In this section we presented an example of using the *Cybernetic Proving Ground*. After the selection of the attack

and choosing parameters, *CPG* deployed a web server and hosts and handled the network settings. Based on the chosen parameters of the experiment, it also successfully orchestrated the attack and captured the outcomes represented by IPFIX flow measurements. Using the measurements we were able to evaluate the attack. The simulation of the attacks verified that the mitigation techniques successfully prevent from this class of attacks. The established testbed can also be used to check resistance of real web servers that can be instantiated in the testbed instead of the artificial web server we used for the experiment.

V. Conclusions

In this paper we have presented a testbed for simulation of cyber attacks. The testbed provides a generic way to simulate and study a wide range of cyber attacks, and facilitates an establishment of isolated virtual environments that researches can use to pursue controlled analysis of attacks. The paper proves the feasibility of the solution, especially it answers the three questions raised in the introduction (real-world environment, flexibility vs. usability and authentic attack modeling).

Using virtualization and clouds we managed to provide an environment where it is possible to configure any common network configuration and therefore we are able to fulfill needs of many kinds of security scenarios. The testbed transparently monitors all components and provides its users with detailed information about activities performed inside the environments as requested by them.

The user can use the *Cybernetic Proving Ground* to set up isolated environments very quickly without the necessity to know details about how to configure networking or deploy auxiliary services like monitoring infrastructure. Instead, users can concentrate solely on the work with the established environment. Being based on a common cloud solution, the framework to establish the environments can be deployed on a wide range of contemporary clouds. The underlying technology also provides sufficient scalability.

We also introduced a concept of security scenarios, which provides a generic way to describe an attack and enables to run its simulation executed in a controlled manner.

Finally, the viability of the solution was demonstrated by a simulation and monitoring of a particular DDoS attack.

Acknowledgements

This work has been supported by the project "Cybernetic Proving Ground" (VG20132015103) funded by the Ministry of the Interior of the Czech Republic.

We appreciate the access to computing facilities *(a)* owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005), and *(b)* provided under the programme Center CERIT Scientific Cloud, part of the Operational Program Research and Development for Innovations, reg. no. CZ. 1.05/3.2.00/08.0144.

References

[1] "Prolexic Quarterly Global DDoS Attack Report Q2 2013," Prolexic Technologies. Accessed on 6 Sep 2013. [Online]. Available: http://www.prolexic.com/knowledge-center-ddos-attack-report-2013-q2.html

[2] "Worldwide Infrastructure Security Report," Arbor Networks. Volume VII, 2012. Accessed on 6 Sep 2013. [Online]. Available: http://pages.arbornetworks.com/rs/arbor/images/WISR2012_EN.pdf

[3] T. Benzel, R. Braden, D. Kim, and C. Neuman, "Experiences With DETER: A Testbed for Security Research," in *Proceedings of the 2nd IEEE Conference on Testbeds and Research Infrastructure for the Development of Networks and Communities (TridentCom)*, 2006.

[4] L. Chen, "Construction of the New Generation Network Security Testbed-Testbed@ TWISC: Integration and Implementation on Software Aspect," 2008, Institute of Computer & Communication, National Cheng Kung University, Tainan, Taiwan.

[5] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An Integrated Experimental Environment for Distributed Systems and Networks," Boston, MA, Dec. 2002, pp. 255–270.

[6] A. Arnes, P. Haas, G. Vigna, and R. A. Kemmerer, "Using a virtual security testbed for digital forensic reconstruction." *Journal in Computer Virology*, vol. 2, no. 4, pp. 275–289, 2007.

[7] B. Van Leeuwen, V. Urias, J. Eldridge, C. Villamarin, and R. Olsberg, "Performing cyber security analysis using a live, virtual, and constructive (LVC) testbed," in *Military Communications Conference 2010 - MILCOM 2010*, 2010, pp. 1806–1811.

[8] K. Krishna, W. Sun, P. Rana, T. Li, and R. Sekar, "V-NetLab: a cost-effective platform to support course projects in computer security," in *Proceedings of 9th Colloquium for Information Systems Security Education*, 2005.

[9] D. Duchamp and G. De Angelis, "A hypervisor based security testbed," in *Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test on DETER Community Workshop on Cyber Security Experimentation and Test 2007*, ser. DETER. Berkeley, CA, USA: USENIX Association, 2007.

[10] Open vSwitch, "Open vSwitch: An Open Virtual Switch," accessed on 30 August 2013. [Online]. Available: http://openvswitch.org/

[11] A. Jeffree, P. Congdon, S. Haddock *et al.*, "Media Access Control (MAC) Bridges and Virtual Bridge Local Area Networks," IEEE Std 802.1Q^{TM}2011, August 2011.

[12] P. Velan, T. Jirsik, and P. Čeleda, "Design and Evaluation of HTTP Protocol Parsers for IPFIX Measurement," in *Advances in Communication Networking, Lecture Notes in Computer Science, Vol. 8115*, T. Bauschert, Ed. Heidelberg: Springer Berlin / Heidelberg, 2013, pp. 136–147.

[13] P. Čeleda, P. Velan, M. Rabek, R. Hofstede, and A. Pras, "Large-Scale Geolocation for NetFlow," in *IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. Ghent, Belgium: IEEE Xplore Digital Library, 2013, pp. 1015–1020.

[14] B. Claise, "Cisco Systems NetFlow Services Export Version 9," RFC 3954 (Informational), Internet Engineering Task Force, 2004.

[15] ——, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information," RFC 5101 (Proposed Standard), Internet Engineering Task Force, 2008.

[16] M. Vizváry and J. Vykopal, "RdpMonitor - RDP authentication attack detection plugin," Masaryk University, 2012, accessed on 6 Sep 2013. [Online]. Available: http://www.muni.cz/ics/services/csirt/tools/rdpmonitor

[17] L. Hellemons, L. Hendriks, R. Hofstede, A. Sperotto, R. Sadre, and A. Pras, "SSHCure: A Flow-Based SSH Intrusion Detection System," in *Dependable Networks and Services*, ser. Lecture Notes in Computer Science, R. Sadre, J. Novotný, P. Čeleda, M. Waldburger, and B. Stiller, Eds. Springer Berlin Heidelberg, 2012, vol. 7279, pp. 86–97.

[18] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures," *Computer*, vol. 45, no. 12, pp. 65–72, 2012.

[19] S. Shekyan, "slowhttptest - Application Layer DoS attack simulator," accessed on 8 August 2013. [Online]. Available: http://code.google.com/p/slowhttptest/