Countermeasure Selection in Virtual Network Systems with Network Intrusion Detection

Kotha Niharika[1],Bindhu Madhavi[2]

[1] M.Tech (CSE), MVR College of Engineering & Technology, A.P., India.

[2]Asst.Professor, Dept. of Computer Science & Engineering, MVR College of Engineering & Technology, A.P., India.

Abstract: TheCloud security be prominent amongst the most essential issues that have pulled in a great deal of innovative work exertion in recent years. Especially, assailants can investigate vulnerabilities of a cloud framework and bargain virtual machines to convey further huge scale (Ddos)Distributed Denial-of-service. Ddos threats normally include early stage activities, for example, multistep abuse, low recurrence weakness checking, and trading off recognized defenseless virtual machine as zombies, lastly Ddos threats through the traded off zombies. To keep powerless virtual machines from being traded off in the cloud, we propose multiphase appropriated weakness recognition, estimation, and counter measure determination system called NICE. The proposed skeleton influences Open Flow system programming Apis to manufactured a screen and control plane over disseminated programmable virtual switches to fundamentally enhance threat discovery and relieve threat results. The framework and security assessments show the effectiveness and adequacy of the proposed result.

Index terms:network security, intrusion detection, attack graph,cloud computing, zombie detection

1INTRODUCTION

The current Cloud Security Alliance (CSA) study demonstrates that among all security issues, ill-use and loathsome utilization of distributed computing is considered as the top security risk, in which aggressors can abuse vulnerabilities in mists and use cloud framework assets to convey threats. In customary server farms, where framework managers have full control over the host machines, vulnerabilities might be identified and fixed by the framework head in an unified way. The test is to create a compelling weakness/threat identification and reaction framework for faultlessly recognizing attacks and minimizing the effect of security rupture to cloud users[1].

By and large, NICE incorporates two principle stages: 1) de-ploy a lightweight reflecting based system interruption discovery executor (NICE-An) on each one cloud server to catch and break down cloud movement. A NICE-An intermittently checks the virtual framework vulnerabilities inside a cloud server to create Scenario Attack Graph (Sags), and afterward focused around the seriousness of distinguished weakness at the community threat objectives, NICE will choose whether or not to put a VM in system examination state. 2) Once a VM enters investigation state, Deep Packet Inspection (DPI) is connected, and/or virtual net-work reconfigurations could be conveyed to the examining VM to make the potential threat practices unmistakable

NICE altogether propels the current system IDS/IPS results by utilizing programmable virtual systems administration approach that permits the framework to develop an element reconfigurable IDS framework. By utilizing programming exchanging strategies, NICE develops a reflecting based movement catching skeleton to minimize the impedance on clients' activity contrasted with conventional knock in-the-wire (i.e., substitute based) IDS/IPS [2,3]. The virtual networking architecture of NICEenables the cloud to establish inspection and quarantine modes for suspicious Vms as indicated by their current powerlessness state in the current SAG. In light of the aggregate conduct of Vms in the SAG, NICE can choose fitting activities, for instance, DPI or movement sifting, on the suspicious Vms.

The range of catching malignant conduct has been decently investigated. This work concentrates on the location of bargained machines that have been enrolled to serve as spam zombies. Their methodology, SPOT, is focused around successively filtering out-going messages while utilizing a measurable system Sequential Probability Ratio Test (SPRT), to rapidly figure out if a host has been traded off. Bothunter identifies bargained machines focused around the way that an exhaustive malware disease process has various overall characterized stages that al-low corresponding the interruption alerts activated by inbound movement with coming about cordial correspondence designs. Botsniffer adventures uniform spatial-transient conduct qualities of bargained machines to identify zombies by gathering streams as indicated by server associations and scanning for comparative conduct in the stream.

Existingsystem: We must note that the configuration of NICE does not mean to enhance any of the current interruption discovery calculations; in-deed, NICE utilizes a reconfigurable virtual systems administration methodology to discover and counter the endeavors to bargain Vms, in this manner anticipating zombie Vms. Decent consolidates a delicate product exchanging answer for isolate and review suspicious Vms for further examination and assurance. Through programmable system approaches, NICE can enhance the assault location likelihood and enhance the flexibility to VM abuse assault without intruding on existing ordinary cloud services [4].

Proposed system: In this paper, we propose Resolving security issues in virtual system frameworks (NICE) to build a protection inside and out interruption discovery schema. For better threat identification, NICE joins threat chart explanatory techniques into the interruption discovery forms. We propose a multistage disseminated helplessness recognition, estimation, and countermeasure determination component called NICE, which is based on threat chart based diagnostic models and reconfigurable virtual system based countermeasures. The proposed skeleton powers Open Flow system programming Apis to manufacture a screen and control plane over disseminated programmable virtual switches to essentially enhance threat location and moderate threat outcomes. The framework and security assessments exhibit the proficiency and adequacy of the proposed result.

NICE MODELS

In this segment, we portray how to use threat diagrams to-model security dangers and vulnerabilities in a virtual organized framework, and propose a VM insurance model focused around virtual system reconfiguration methodologies to keep Vms from being misused.

Threat model

In our threat model, we accept that an aggressor could be spotted either outside or within the virtual systems

administration framework. The aggressor's essential objective is to endeavor helpless Vms and com-swear up and down to them as zombies. Our proposed result could be sent in an Infrastructure-as-a-Service (Iaas) cloud organizing framework, and we expect that the Cloud Service Provider (CSP) is considerate. We likewise expect that cloud administration clients are allowed to introduce whatever working frameworks or applications they need, regardless of the possibility that such activity may acquaint vulnerabilities with their controlled Vms. Physical security of cloud server is out of degree of this paper. We acknowledge that the hypervisor is secure and free of any weakness. The issue of a malevolent inhabitant breaking out of Domu and getting access to physical server have been considered in late work [21] and are out of degree of this paper.

Attack graph model

A threat chart is a displaying instrument to represent all conceivable multistage, multihost threat ways that are critical to under-stand dangers and afterward to choose proper countermeasures [22]. In an threat diagram, every hub speaks to either precondition or outcome of an endeavor. The activities are not so much a dynamic threat on the grounds that typical convention collaborations can additionally be utilized for threats. Threat chart is useful in distinguishing potential dangers, conceivable threats, and known vulnerabilities in a cloud framework. Since the threat diagram gives de-tails of all known vulnerabilities in the framework and the integration data, we get an entire picture of current security circumstance of the framework, where we can foresee the conceivable dangers and threats by corresponding discovered occasions or exercises.In the event that an occasion is perceived as a potential assault, we can apply particular countermeasures to moderate its effect or take activities to keep it from tainting the cloud framework. To speak to the assault and the aftereffect of such activities, we broaden the documentation of Mulval rationale assault diagram as displayed by Ou et al. [12] and characterize as Scenario Attack Graph (SAG).

We expect that A contains accumulated alarms as opposed to crude cautions. Crude cautions having same source and goal IP advertisement dresses, assault sort, and time stamp inside a pointed out window are accumulated as Meta Alerts. Each one requested pair ða; a0þ in ACG maps to two neighbour vertices in SAG with time stamp distinction of two cautions inside a predefined limit. ACG shows reliance of cautions in ordered request and we can discover related alarms in the same assault situation via looking the alarm way in ACG. A set P is utilized to store all ways from root alarm to the target caution in the SAG, and every way Si _ P speaks to alarms that fit in with the same assault situation. We clarify a system for using ACG together to anticipate an aggressor's conduct. Alarm Correlation calculation is taken after for each caution discovered and gives back one or more ways Si. For each alarm air conditioning that is gotten from the IDS, it is added to ACG in the event that it doesn't exist. For this new caution air conditioning, the comparing vertex in the SAG is found by utilizing capacity mapðacþ (line 3). For this vertex in SAG, caution identified with its parent vertex of sort NC is then associated with the mongrel rent alarm air conditioning (line 5). This makes another set of alarms that have a place with a way Si in ACG (line 8) or parts out another way Siþ1 from Si with subset of Si before the caution an and annexes air conditioning to Siþ1 (line 10). Toward the end of this calculation, the ID of air conditioning will be added to caution quality of the vertex i.

Algorithm 1 returns a set of attack paths S in ACG.

Algorithm. AlertCorrelation
need alert ac, SAG, ACG
Step 1: if ac then (ac is a new alertthen)
Step 2:create node ac in ACG
Step 3: n1 vc 2 mapðacÞ
Step 4:for all n2 2 parentðn1Þ do
Step 5:create edge (n2:alert; ac)
Step 6: for all Si containing a do
Step 7: if a is the last element in Si
Step 8: then
Step 9: append ac to Si
Step 10: else
Step 11:create path Siþ1 ¼ fsubsetðSi; aÞ; acg
Step 12:end if
Step 13: end for add ac to n1:alert
Step 14:end for
Step 15: end if
Step 16:return S

VM protection model

The VM insurance model of NICE comprises of a VM profiler, a security indexer, and a state screen. We determine security file for all the Vms in the system relying on different variables like integration, the quantity of vulnerabilities present and their effect scores. The effect score of vulnerability as characterized by the CVSS guide [24], serves to judge the privacy, respectability, and accessibility effect of the defencelessness being abused. Network metric of a VM is chosen by assessing approaching and cordial associations.

Definition (VM State): Based on the data accumulated from the system controller, VM states could be characterized as taking after:

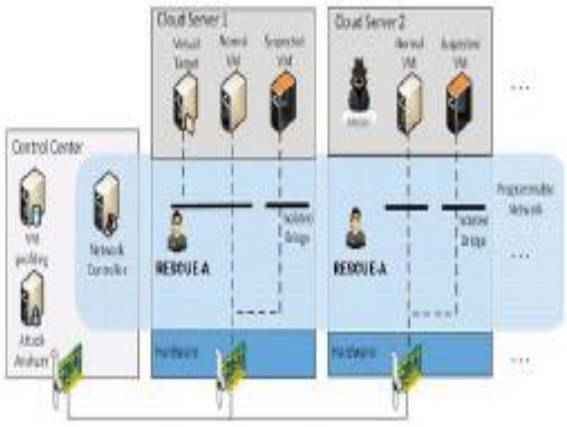
1. Stable. There does not exist any known weakness on the VM.
2. Vulnerable. event of one or more vulnerabilities on a VM, which stays unexploited.
3. Exploited. No less than one defencelessness has been misused and the VM is bargained.
4. Zombie. VM is under control of aggressor.

NICE SYSTEM DESIGN

In this segment, we first present the framework outline review of Pleasant and after that itemized depictions of its parts.

System outline review

The proposed NICE schema is shown in Fig. 1. It demonstrates the NICE system inside one cloud server group. Real parts in this schema are circulated and light weighted NICE-An on every physical cloud server, a system controller, a VM profiling server, andan assault analyzer.The recent three segments are found in an incorporated control focus joined with softwareswitches on each cloud server (i.e., virtual switches built on one or multiple Linux software bridges). NICE-A is a software agent implemented in each cloud server connected to the control center through a dedicated and isolated secure channel, which is separated from the normal data packets using OpenFlow tunneling or VLAN approaches. The network controller is responsible for deploying attack countermeasures based on decisions made by the attack analyzer.



RESCUE Architecture with single cloud cluster

Fig. 1.*NICE Architecture with single cloud cluster*

In the following description, our phrasings are focused around the XEN virtualization innovation. Pleasant A will be a system interruption identification motor that could be introduced in either Dom0 or Domu of a XEN cloud server to catch and channel malignant movement. Interruption recognition cautions are sent to control focus when suspicious or bizarre activity is distinguished. In the wake of getting a caution, assault analyzer assesses the seriousness of the alarm focused around the assault chart, chooses what counter measure methods to take, and after that launches it through the system controller. An assault chart is made as per the weakness data inferred from both logged off and continuous powerlessness examines. Disconnected from the net checking is possible by running entrance tests and online ongoing helplessness separating could be enacted by the framework controller (e.g., when new ports are opened and recognized by Ofss) or when new cautions are produced by the NICE-A. Once new vulnerabilities are found or countermeasures are conveyed, the assault chart will be remade. Countermeasures are started by the assault analyzer focused around the assessment results from the expense profit investigation of the viability of countermeasures. At that point, the system controller launchs countermeasure activities by reconfiguring virtual or physical Ofss.

System components:

In this section, we explain each component of NICE.

a) **NICE-A**

The NICE-A will be a Network-based Intrusion Detection Sys-tem (NIDS) operator introduced in either Dom0 or Domu in each one cloud server. It filters the movement experiencing Linux connects that control all the activity among Vms and in/out from the physical cloud servers. In our examination, Snort is utilized to actualize NICE-An in Dom0. It will sniff a reflecting port on every virtual scaffold in the Open vswitch (OVS). then again, our configuration is autonomous to the introduced VM. In the execution assessment segment, we will exhibit the tradeoffs of introducing RES-CUE-An in Dom0 and Domu. We must note that the alarm discovery nature of NICE-A relies on upon the execution of NICE-A, which uses Snort. We don't concentrate on the recognition precision of Snort in this paper. Subsequently, the individual alarm discovery's false caution rate does not change. Then again, the false caution rate could be diminished through our structural engineering outline. We will talk about all the more about this issue in the later segment.

b) **VM Profiling**

Virtual machines in the cloud might be profiled to get exact data about their state, administrations running, open ports, etc. One central point that tallies to a VM profile is its integration with different VMs. Any VM that is joined with more number of machines is more vital than the one associated with less VMs on the grounds that the impact of trade off of a profoundly joined VM can result in more harm. Additionally needed is the learning of administrations running on a VM to confirm the realness of cautions relating to that VM. An aggressor can utilize port-examining system to perform an exceptional examination of the system to search for open ports on any VM. So data about any open ports on a VM and the historical backdrop of opened ports

assumes a critical part in deciding how defenceless the VM is. All these variables consolidated will structure the VM profile. VM profiles are kept up in a database and contain exhaustive data about vulnerabilities, alarm, and activity.

c) **Attack Analyzer**

The significant capacities of NICE framework are performed by at-tack analyzer, which incorporates methodology, for example, assault chart development and redesign, caution association, and countermeasure choice. The procedure of building and using the SAG comprises of three stages: Information get-together, assault chart development, and potential adventure way investigation. With this data, assault ways could be displayed utilizing SAG. Every hub in the assault chart speaks to an adventure by the aggressor.

Each path from an initial node to a goal node represents a successful attack. In summary, NICE attack graph is constructed based on the following information:

• Cloud system information is collected from the node controller (i.e., Dom0 in XenServer). The information includes the number of VMs in the cloud server, running services on each VM, and VM's Virtual Interface (VIF) information.

• Virtual network topology and configuration information is collected from the network controller, which includes virtual network topology, host connectivity, VM connectivity, every VM's IP address, MAC address, port information, and traffic flow in-formation.

• Vulnerability information is generated by both on demand vulnerability scanning (i.e., initiated by the network controller and NICE-A) and regular penetration testing using the well-known vulnerability databases, such as Open Source Vulnerability Database (OSVDB) [20], Common Vulnerabilities and Exposures List (CVE) [22], and NIST National Vulnerability Database (NVD) [23].

The attack analyzer also handles alert correlation and analysis operations. This component has two major functions:

1) constructs ACG, and 2) provides threat information

and appropriate countermeasures to network controller for virtual network reconfiguration. After receiving an alert from NICE-A, alert analyzer matches the alert in the ACG. If the alert already exists in the graph and it is a known attack (i.e., matching the attack signature), the attack analyzer performs countermeasure selection procedure according to the algorithm describe and then notifies network controller immediately to deploy countermeasure or mitigation actions. If the alert is new, at-tack analyzer will perform alert correlation and analysis ac-cording to Algorithm 1, and updates ACG and SAG. This algorithm correlates each new alert to a matching alert correlation set (i.e., in the same attack scenario). A selected counter-measure is applied by the network controller based on the severity of evaluation results. If the alert is a new vulnerability and is not present in the NICE attack graph, the attack analyzer adds it to attack graph and then reconstructs it.

d) **Network Controller**

The network controller is a key component to support the programmable networking capability to realize the virtual network reconfiguration feature based on OpenFlow protocol [20]. In NICE, within each cloud server there is a software switch, for example, OVS [5], which is used as the edge switch for VMs to handle traffic in and out from VMs. The communication between cloud servers (i.e., physical servers) is handled by physical OpenFlow-capable Switch (OFS). In NICE, we integrated the control functions for both OVS and OFS into the network controller that allows the cloud system to set security/filtering rules in an integrated and comprehensive manner.

The system controller is in charge of gathering system data of current Open Flow system and gives info to the threat analyzer to develop threat charts. Through the cloud inside disclosure modules that utilization DNS, DHCP, LLDP, and stream starts [27], system controller can find the system network data from OVS and OFS. This data incorporates current information ways on each one switch and itemized stream data connected with these ways, for example, TCP/IP and MAC header. The system stream and topology change data will be naturally sent to the controller and afterward conveyed to threatanalyzer to recreate threat diagrams. An alternate essential capacity of the system controller is to support the threat analyzer module. As indicated by the Openflow convention [20], when the controller gets the first parcel of a stream, it holds the bundle and checks the stream table for going along activity strategies. In NICE, the system control likewise counsels with the threat analyzer for the stream access control by setting up the separating leads on the relating OVS and OFS. When an activity stream is conceded, the accompanying parcels of the stream are not taken care of by the system controller, however checked by the NICE-A.

NICEATTACK IMPROVEMENT AND COUNTER MEASURES

In this segment, we display the strategies for selecting the countermeasures for a given assault situation. At that point vulnerabilities are found or a few Vms are distinguished as suspicious, a few countermeasures could be taken to limit aggressors' capacities and it is paramount to separate in the middle of bargained and suspicious Vms. The countermeasure serves the purpose of: 1) protecting

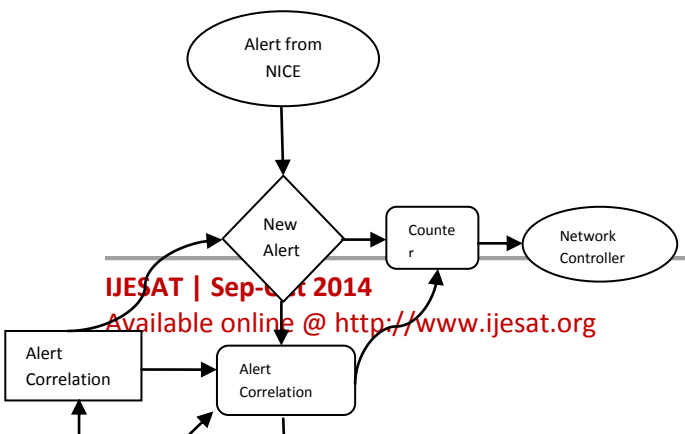


Fig. Attack analyzer for Workflow.

the target VMs from being compromised, and 2) making at-tack behaviour stand prominent so that the attackers' actions can be identified.

IMPROVEMENT Strategy

In view of the security measurements characterized in the past subsection, NICE can develop the relief techniques because of discovered alarms. To start with, we characterize the term counter-measure pool as takes after:

Definition 2 (Countermeasure Pool). A countermeasure pool

CM ¼ fcm1; cm2; . . . ; cmng is a set of countermeasures. Eachcm 2 CM is a tuple cm = (cost, intrusiveness, condition, effectiveness).

In general, there are many countermeasures that can be applied to the cloud virtual networking system depending on available countermeasure techniques that can be applied. Without losing the generality, several common virtual-networking-based countermeasures are listed in Table 1. The optimal countermeasure selection is a multiobjective optimization problem, to calculate MIN (impact, cost) and MAX (benefit).

In NICE, the network reconfiguration strategies mainly involve two levels of action: Layer-2 and layer-3. At layer-2, virtual bridges (including tunnels that can be established between two bridges) and VLANs are main component in cloud's virtual networking system to connect two VMs directly. A virtual bridge is an entity that attaches VIFs. Virtual machines on

different bridges are isolated at layer 2. VIFs on the same virtual bridge but with different VLAN tags cannot communicate to each other directly. Based on this layer-2 isolation, NICE can deploy layer-2 network reconfiguration to isolate suspicious VMs. For example, vulnerabilities due to Arpspoofing [22] attacks are not possible when the suspicious VM is isolated to a different bridge. As a result, this counter-measure disconnects an attack path in the attack graph causing the attacker to explore an alternate attack path. Layer-3 reconfiguration is another way to disconnect an attack path. Through the network controller, the flow table on each OVS or OFS can be modified to change the network topology.

We must note that utilizing the virtual system reconfiguration approach at lower layer has the focal point in that upper layer applications will encounter insignificant effect. Particularly, this methodology is just conceivable when utilizing programming exchanging methodology to computerize the reconfiguration in a very dynamic systems nature's domain. Countermeasures, for example, activity disconnection could be executed by using the traffic engineering abilities of OVS and OFS to confine the limit and reconfigure the virtual system for a suspicious stream.

CONCLUSION

In this paper, we discussed NICE, which is proposed to recognize and moderate communitarian threats in the cloud virtual systems nature. Pleasant uses the threat diagram model to lead threat location and forecast. The master postured result researches how to utilize the programmability of programming switches-based answers for enhance the identification precision and annihilation exploited person abuse periods of shared threats. The framework execution assessment shows the plausibility of NICE and demonstrates that the proposed result can fundamentally decrease the danger of the cloud framework from being misused and ill-used by inside and outside assailants.

Pleasant just researches the system IDS methodology to counter zombie explorative threats. To enhance the identification accuscandalous, host-based IDS results are required to be consolidated and to blanket the entire range of IDS in the cloud framework. This ought to be explored later on work. Moreover, as showed in the paper, we will examine the versatility of the proposed NICE result by exploring the decentralized system control and threat dissection model focused on current study.

REFERENCES

[1] “Securing Cloud ComputingEnvi-ronment Against DDoS Attacks,”B. Joshi, A. Vijayan, and B. Joshi, Proc. IEEE Int’l Conf.Computer Comm. and Informatics (ICCCI ’12), Jan. 2012.

[2] “Security and PrivacyChallenges in Cloud Computing Environments,” H. Takabi, J.B. Joshi, and G. Ahn, IEEE Securityand Privacy, vol. 8, no. 6, pp. 24-31, Dec. 2010.

[3] “Detecting Spam Zombies by Monitoring OutgoingMessages,” Z. Duan, P. Chen, F. Sanchez, Y. Dong, M. Stephenson, and J.Barker, IEEE Trans. Dependable and Secure Computing, vol. 9,no. 2, pp. 198-210, Apr. 2012.

[4]“BotHunter: Detecting Malware Infection through IDS-drivenDialog Correla-tion,” G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee , Proc. 16th USENIX Security Symp. (SS ’07),pp. 12:1-12:16, Aug. 2007.

[5]"BotSniffer: Detecting BotnetCommand and Control Channels in Network Traffic," G. Gu, J. Zhang, and W. Lee, Proc. 15thAnn. Networkand Distributed Sytem Security Symp. (NDSS '08), Feb.2008.

[6]"Automated Generation and Analysis of Attack Graphs," O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J.M. Wing Proc.IEEE Symp. Security and Privacy, pp. 273-284, 2002.

[7] "Scalable, graphbasednetwork vulnerability analysis," P. Ammann, D. Wijesekera, and S. Kaushik, Proc. 9th ACM Conf.Computer and Comm. Security (CCS '02), pp. 217-224, 2002.

[8] "MulVAL: A Logic-Based Network Security Analyzer,"X. Ou, S. Govindavajhala, and A.W. Appel, Proc. 14th USENIX SecuritySymp., pp. 113-128, 2005.

[9] "Alert Correlation Survey: Frameworkand Techniques," R. Sadoddin and A. Ghorbani,Proc. ACM Int'l Conf. Privacy, Security andTrust: Bridge the Gap between PST Technologies and Business Services(PST '06), pp. 37:1-37:10, 2006.

[10] "Scalable Optimal CountermeasureSelection Using Implicit Enumeration on Attack CountermeasureTrees,"A. Roy, D.S. Kim, and K. Trivedi, Proc. IEEE Int'l Conf. Dependable Systems Networks(DSN '12), June 2012.

[11] "Dynamic Security RiskMan-agement Using Bayesian Attack Graphs,"N. Poolsappasit, R. Dewri, and I. Ray, IEEE Trans.Dependable and Secure Computing, vol. 9, no. 1, pp. 61-74, Feb. 2012.

[12] "Software-Defined Networking:The New Norm for Networks," Open Networking Fundation, ONF White Paper, Apr. 2012.

[13] "OpenFlow:Enabling Innovation in Campus Networks," N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L.Peterson, J. Rexford, S. Shenker, and J. Turner, SIGCOMM ComputerComm. Rev., vol. 38, no. 2, pp. 69-74, Mar. 2008.

[14] E. Keller, J. Szefer, J. Rexford, and R.B. Lee, "NoHype:

[15] " A Scalable Approach toAttack Graph Generation," X. Ou, W.F. Boyer, and M.A. McQueen, Proc. 13th ACM Conf. Computer andComm. Security (CCS '06), pp. 336-345, 2006.

[16] O. Database, "Open Source Vulnerability Database (OVSDB),"http://osvdb.org/, 2012.

[17] National Institute of Standards and Technology, "NationalVulner-ability Database, NVD," http://nvd.nist. gov, 2012.

[18] X. Ou and A. Singhal, Quantitative Security Risk Assessment ofEn-terprise Networks. Springer, Nov. 2011.

[19] "Measuring Network Security UsingBayesian Network-Based Attack Graphs,"M. Frigault and L. Wang, Proc. IEEE 32nd Ann.Int'l Conf. Computer Software and Applications (COMPSAC '08),pp. 698-703, Aug. 2008.

[20] "Network Security ManagementUsing ARP Spoofing," K. Kwon, S. Ahn, and J. Chung,Proc. Int'l Conf. Computational Science and ItsApplications (ICCSA '04), pp. 142-149, 2004.

[21]"Armitage,"http://www.Fastandeasyhacking.com, 2012.

[22] "VEA-bility Security Metric:A Network Security Analysis Tool," M. Tupper and A. Zincir-Heywood, Proc. IEEE Third Int'l Conf.Availability, Reliability and Security (ARES '08), pp. 950-957, Mar.2008.