International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Graph-based Attack Detection in Cloud using KDD CUP 99 Dataset

Swapnali G. Game[1], S. B. Natikar[2]

[1, 2]Department of Computer Engineering, VACOE, Ahmednagar, University of Pune, India

Abstract: ***In the area of research and development effort for cloud computing, Cloud security is considered as one of challenging issues. Most commonly faced attacks are Distributed Denial-of-Service (DDoS) attacks. DDoS attacks are variation of DOS attack at distributed and large-scale level. Firstly attacker tries to discover the vulnerabilities or we can say loopholes of a cloud system and takes control over the virtual machines. And then gets success in deploying DDoS at large scale. Such attacks includes certain actions at initial stage such as exploitation in multiple steps, scanning for uncommon or less occurring vulnerabilities, identified vulnerabilities are utilized against virtual machines to use them as zombies and finally DDOS is achieved through these compromised zombies. To avoid vulnerable virtual machines from being compromised in the cloud system, proposed approach uses multiphase vulnerability detection at distributed level, measurement, countermeasure selection mechanism called as NICE, which is based on attack graph based models and reconfigurable virtual network based countermeasures. Use of standard dataset KDD Cup 99 dataset helps to cover most of the types of intrusion signatures and features. There is a need of processing encrypted traffic also together with plain traffic flowing through the cloud system. As included in the proposed system, host-based intrusion detection system implementation gives more benefits as compare to NIDS based implementation.***

Keywords*:* DDos, IDS, Attack Graph, KDD Cup 99 Dataset, Zombie

1. Introduction

Recent studies have shown that users migrating to the cloud consider security as the most important factor. A recent Cloud Security Alliance (CSA) survey shows that among all security issues, abuse and nefarious use of cloud computing is considered as the top security threat [2], in which attackers can exploit vulnerabilities in clouds and utilize cloud system resources to deploy attacks. In traditional data centers, where system administrators have full control over the host machines, vulnerabilities can be detected and patched by the system administrator in a centralized manner. However, patching known security holes in cloud data centers, where cloud users usually have the privilege to control software installed on their managed VMs, may not work effectively and can violate the service level agreement (SLA). Furthermore, cloud users can install vulnerable software on their VMs, which essentially contributes to loopholes in cloud security. The challenge is to establish an effective vulnerability/attack detection and response system for accurately identifying attacks and minimizing the impact of security breach to cloud users.

The aim of Intrusion Detection is Detecting and reacting to an attack. But the current solution by IDS as well as firewall does not work very well in real life. For any IDS implementations the large volume of raw alerts from IDS and false alarms are two major problems. For signature-based IDSs (one of the method of IDS) there will be gag between a new threat discovery and its signature being used by the IDS. But in meanwhile the IDS will be unable to identify the threat whose signature is not available with current IDS.

It is usual practice to implement a firewall or a security policy, but experience has shown IDS and firewall both as alone are dramatically insufficient as we can see the current security scenarios in IT world.

2. Literature Survey

Now we discuss literatures of several highly related research areas to IDS, Zombie detection and prevention techniques, their drawbacks.

A considerable amount of research has been done towards detecting malicious behavior. Here different methods and techniques are discussed. Detecting malicious behavior has been well explored by Z. Duan, P. Chen, F. Sanchez, Y. Dong, M. Stephenson, and J. Barker [3] discuss detection of compromised machines that have been chosen to serve as spam zombies. Their approach called as SPOT, is based on sequentially scanning outgoing messages while employing a statistical method Sequential Probability Ratio Test (SPRT), to quickly determine whether a host has been compromised or not. Antonio Bianchi, Yan Shoshitaishvili, Christopher Kruegel, Giovanni Vigna [4] detect compromised machines by comparing images of physical memory taken from similar machines to identify differences associated with rootkit infections. G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee [5] detect compromised machines through malware infection process which has a number of well-defined stages that allow correlating the intrusion alarms triggered by inbound traffic with resulting outgoing communication patterns. G. Gu, J. Zhang, and W. Lee [6] first exploit uniform spatial-temporal behavior characteristics of compromised machines and then detect zombies by grouping flows by considering server connections and searching for similar behavior in the flow.

Each path in an attack graph is a series of exploits or actions that leads to an undesirable state. We can say an undesirable state is a state where the intruder has obtained administrative access to a critical host. Firstly, scanning tools determine vulnerabilities of individual hosts. Then the analyst produces an attack graph using local vulnerability information along with other information about the network like connectivity between hosts [7]. There are many automation tools are available to construct attack graph. O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J.M. Wing [8] constructs attack graph using a technique based on Binary Decision Diagrams (BDDs) and modified symbolic model checking NuSMV [9]. Although their model can generate all possible attack paths, the scalability is a big issue for their solution. P. Ammann, D. Wijesekera, and S. Kaushik [10] considered the assumption of monotonicity, which explains that the precondition of a given exploit is never invalidated by the successful application of another exploit. It means attackers never need to backtrack. With this assumption, it becomes possible to obtain a scalable and concise graph representation for encoding attack tree. X. Ou, S. Govindavajhala, and A.W. Appel [11] present MulVAL (Multihost, multistage Vulnerability Analysis), a framework for modelling the interaction of software bugs with system and network configurations. MulVAL uses Datalog as its modelling language. MulVAL comprises a scanner which tests a machine for vulnerable software. MulVAL aims at detecting potential attack paths before an attack happens. But to provide the security assessment and alert correlation features it is possible to modify and extend MulVAL's attack graph structure [1]. Kyle Ingols, Matthew Chu, Richard Lippmann, Seth Webster, Stephen Boyer [12] describe substantial enhancements to the NetSPA attack graph system required to model additional present-day threats and countermeasures like host-based vulnerability scans, intrusion prevention systems, proxy firewalls and personal firewalls.

The aim of Intrusion Detection is Detecting and reacting to an attack. But the current solution by IDS as well as firewall does not work very well in real life. For any IDS implementations the large volume of raw alerts from IDS and false alarms are two major problems. For signature-based IDSs (one of the method of IDS) there will be gag between a new threat discovery and its signature being used by the IDS. But in meanwhile the IDS will be unable to identify the threat whose signature is not available with current IDS. Alert correction tool plays an important role in identify the source or target of the intrusion in the network and also specially to detect multistep attack.

Many attack graph-based alert correlation techniques have been proposed recently. S. Roschke, F. Cheng, and C. Meinel [13] proposed an AG based correlation algorithm that overcomes the limitations in applying the nested loop-based correlation methods and proposed a QG called queue graph approach to remove this limitation. The algorithm is able to identify multiple attack scenarios of the same anatomy by using an attack graph. Once any exploit is examined QG is used to trace alerts matching each exploit in the attack graph. But the algorithm needs some computing power to consume and algorithm needs to be tested using larger data sets. L. Wang, A. Liu, and S. Jajodia [14] extend the basic QG approach to a unified method to hypothesize missing alerts and to predict future alerts and propose a compact representation for the result of alert correlation. But the limitations of this method are overcome in [13].

Once the possible attack scenarios are known, selecting and then applying countermeasure is the next important step. Selecting optimal countermeasures depends on attack path and cost benefit analysis so that final solution cost can be optimal as much as possible. N. Poolsappasit, R. Dewri, and I. Ray [15] proposed a Bayesian attack graph (BAG) model of the network which enables to better understand the causal relationships between pre-conditions, vulnerability exploitations, and post-conditions. They proposed a genetic algorithm capable of performing both single and multi-objective optimization of the system administrator's objectives. Using a BAG, the system administrator performs risk assessment and risk mitigation and uses genetic algorithm for giving solution to the countermeasure optimization problem. A. Roy, D.S. Kim, and K. Trivedi [16] proposed an attack countermeasure tree (ACT) which is considering both attacks and its countermeasures. They used greedy and branch and bound techniques to minimizing the number of countermeasures. This approach aims for minimizing security investment cost and maximizing the benefit from implementing a certain countermeasure set in the ACT.

In implemented system, while implementing the protection model, we are having both the options host-based IDS or network-based IDS. HIDS is appropriate for protecting an individual computer systems and the information it contains as the name itself indicates. However it doesn't provide data security on the network as a whole. Also the security systems take on considerable processing resource of the host like RAM, CPU and storage. NIDS monitor and analyze network traffics on a designated network segment. It can be categorized as knowledge or behavior based. But for knowledge based NIDS, system can generate few false positives, good packets are labelled as bad packets and transmission could be interrupted due to poorly defined signature one more important thing to consider is NIDS is unable to stop encrypted packets of system attack from intruders. To overcome the limitations of HIDS or NIDS, it is possible to combine the strength of both systems by forming hybrid systems that is HIDS. But we need to work for reducing false alarms and the large volume of raw alerts generated by IDS system. So, as per need NICE is implemented with host based IDS.

Multiple tools are available in the market for handling functions of constructing attack graphs, updating attack graphs, selecting optimal cost countermeasures then finally applying selected countermeasures successfully by reducing overall solution cost.

There are plenty of IDS tools available in the market, for example SNORT & BRO are popular IDS systems available in the market. But Bro is the suitable IDS system for users those are UNIX experts. It plays supporting role to the main IDS system. Bro is a good choice if user wants to customize IDS according to his/her network. As compared to Snort Bro is more effective for Gbps networks. For high speed networks snort is a good choice. Snort mainly focuses on simplicity and performance accuracy. This is the main plus point which makes the snort best choice to run on any operating system. But main parameter of comparison is false alarm rate which significantly affect the overall system performance.

3. Design Overview

The implemented NICE framework is illustrated in figure 1. It shows the NICE framework within one cloud server cluster. Major components in this framework are distributed and light-weighted NICE-A on each physical cloud server, a network controller, a VM profiling server, and an attack analyzer. The latter three components are located in a centralized control center connected to software switches 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.

In the following description, terminologies are based on the XEN virtualization technology. NICE-A is a network intrusion detection engine that can be installed in either Dom0 or DomU of a XEN cloud server to capture and filter malicious traffic. Intrusion detection alerts are sent to control center when suspicious or anomalous traffic is detected. After receiving an alert, attack analyzer evaluates the severity of the alert based on the attack graph, decides what countermeasure strategies to take, and then initiates it through the network controller. An attack graph is established according to the vulnerability information derived from both offline and real-time vulnerability scans. Offline scanning can be done by running penetration tests and online real-time vulnerability scanning can be triggered by the network controller (e.g., when new ports are opened and identified by OFSs) or when new alerts are generated by the NICE-A. Once new vulnerabilities are discovered or countermeasures are deployed, the attack graph will be reconstructed. Countermeasures are initiated by the attack analyzer based on the evaluation results from the cost-benefit analysis of the effectiveness of countermeasures. Then, the network controller initiates countermeasure actions by reconfiguring virtual or physical OFSs.

Since the attack graph provides details of all known vulnerabilities in the system and the connectivity info, we get an entire picture of current security situation of the system, where we can guess the likely extortions and attacks by correlating detected events or actions. If an incident is recognized as a possible attack, we can apply precise countermeasures to moderate its impact or take actions to prevent it from contaminating the cloud system. To signify the attack and the consequence of such activities, we extended the scheme of MulVAL logic attack graph as presented by X. Ou, S. Govindavajhala, and A.W. Appel [11] and define as Scenario Attack Graph (SAG).

Definition: SAG: An SAG is a tuple SAG = (V, E), where

1) V = NC U ND U NR denotes a set of vertices that include three types namely conjunction node NC to denote exploit, dislocation node ND to denote outcome of exploit, and root node NR for viewing initial step of an attack scenario.
2) E = Epre U Epost denotes the set of directed edges. An edge e € Epre ND × NC represents that ND must be satisfied to achieve NC. An edge e € Epost NC × ND means that the consequence shown by ND can be obtained if NC is satisfied.
   Node vc € NC is defined as a three tuple (Hosts, vul, alert) representing a set of IP addresses, vulnerability information such as CVE [18], and alerts related to vc, respectively. ND behaves like a logical OR operation and contains details of the results of actions. NR represents the root node of the SAG.
   For correlating the alerts, we have referred to the approach described in [13] and defined a new Alert Correlation Graph (ACG) to map alerts in ACG to their respective nodes in SAG. To retain track of attack growth, we track the source and destination IP addresses for attack activities.

Definition: ACG: An ACG is a three tuple ACG = (A, E, P), where

1) A is a set of aggregated alerts. An alert from set A, a € A is a data structure (src, dst, cls, ts) representing first source IP address, second destination IP address, third type of the alert that is generated, and lastly time stamp of the alert respectively.
2) Each alert a maps to a pair of vertices (vc, vd) in SAG using map(a) function, i.e., map(a) : a →{(vc, vd) | (a.src € vc.Hosts) ^ (a.dst € vd.Hosts) ^ (a.cls = vc.vul)}.
3) E is a set of directed edges representing correlation between two alerts (a, a') if criteria below satisfied:
   a. (a.ts < a'.ts) ^ (a'.ts – a.ts < threshold).
   b. ∃(vd, vc) € Epre: (a.dst € vd.Hosts ^ a'.src € vc.Hosts).
4) P is set of paths in ACG. A path Si i.e. subset of P is a set of related alerts in chronological order.

It is assumed that A contains aggregated alerts rather than the raw alerts. Raw alerts having identical destination and source IP addresses, time stamp within a specified window and attack type are aggregated as Meta Alerts. Each ordered pair (a, a') in ACG maps to two neighbor vertices in SAG with time stamp difference of two alerts within a predefined threshold values. ACG demonstrations dependency of alerts in consecutive order and we can find related alerts in the same attack scenario by searching the alert path in Attack Correlation Graph. A set P is mainly used to store all paths from root alert node to the target alert node in the SAG, and each path S_i i.e. subset of P represents alerts that belong to the same attack scenario.

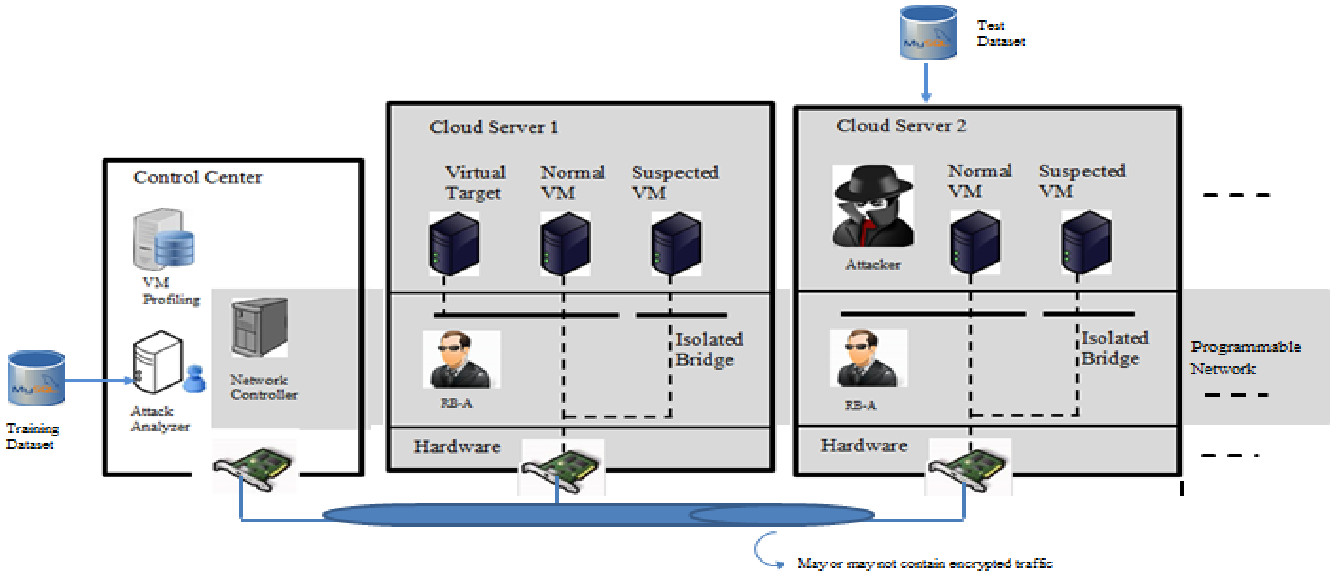

Figure 1: Proposed NICE (RB-A) Architecture

4. System Components

A. RB-A or NICE-A

The NICE-A is a Network-based Intrusion Detection System (NIDS) agent (also called as Request Broker-Agent) installed in either Dom0 or DomU in each cloud server. Main task handled by it is, it scans the traffic passing through Linux bridges that regulate all the traffic among VMs and in as well as out from the physical cloud servers. In the implemented system, to implement NICE-A in Dom0, Snort is used.

B. VM Profiling

Virtual machines in the cloud can be profiled to get precise information related to their state, open ports, services running, and so on. One key factor that counts toward a VM profile is its connectivity with other virtual machines. Also required is the knowledge of services running on a VM so as to verify the authenticity of alerts related to that VM. Port-scanning program can be used by an attacker to perform detail inspection of the network to find open ports on any VM. Information about any open ports on a VM and the history of ports those are opened plays important role in determining how vulnerable the VM is. By combining all these factors will form the VM profile.

C. Attack Analyzer

The key functions of NICE system are done by attack analyzer, including procedures such as attack graph construction and update, alert correlation, and countermeasure selection. The process of constructing and utilizing the SAG consists of three phases: 1. Information gathering, 2. Attack graph construction, and 3. potential exploit path analysis. By using this information, attack paths can be demonstrated using SAG. Each node in the attack graph represents an exploit by an attacker. Every single path from an initial node to a goal node denotes a successful attack. The attack analyzer also handles alert correlation and analysis operations. It is having two main functions: 1. Constructs ACG, 2. Provides threat information and appropriate countermeasures to network controller for virtual network reconfiguration.

D. Network Controller

Network controller is also responsible for applying the countermeasure from attack analyzer. Countermeasures are selected by NICE based on severity of an alert and VM Security Index (VSI), and executed by the network controller. In case of a severe alert is generated and finds some known attacks, or a VM is noticed as a zombie, the network controller will block the VM immediately.

E. KDD Cup 99 dataset

A well-known data mining competition called KDD Cup is the annual ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD cup set is the data set used 1999 KDD intrusion detection contest. This dataset is defined by Stolfo et al. and is designed based on the data captured in 1998 DARPA Intrusion Detection Evaluation Program by MIT Lincoln Labs called as DARPA'98. DARPA'98 is 4 gigabytes of compressed raw tcpdump data of duration of 7 weeks of network traffic, which can be processed into about 5 million connection records, each with near about 100 of bytes. KDD training dataset consists of around 4,900,000 single connection vectors (record) where each of which contains 41 features (column) and is labeled as either normal or an attack, with exactly one specific attack type [17].

5. Result Analysis

We evaluate system performance to provide guidance on how much traffic NICE can handle for one cloud server and use the evaluation metric to scale up to a large cloud system. To demonstrate the feasibility of the implemented work, comparative studies were conducted with several parameters like bandwidth utilization, no. of packets used, time required to handle no. of packets etc.

We are also considering system performance in both traffic capturing mechanism mirror-based traffic capturing mechanism and proxy based traffic capturing mechanism. Again true positive and false positive probability in both type of traffic capturing mechanism is also considered.

For evaluating the NICE system's performance we have tested the system with the help of five nodes and according to their response we got the following graphs of result or performance analysis.

Probability of True positive and False positive alert detection in NICE, Mirror-based and Proxy-based traffic capturing mechanisms

First we see what is TP and FP,

- True positive (TP): The amount of attack detected when it is actually attack.
- False positive (FP): The amount of attack detected when it is actually normal called as false alarm.

In this scenario X-axis shoes the no. of packets and Y-axis shows the detection rate of intrusions then it may False positive (FP) or True positive (TP).

After observing the following graphs, we can say that the performance of the NICE system as compare to remaining two approaches is good. Because probability of TP get increases as the no. of packets get increased.

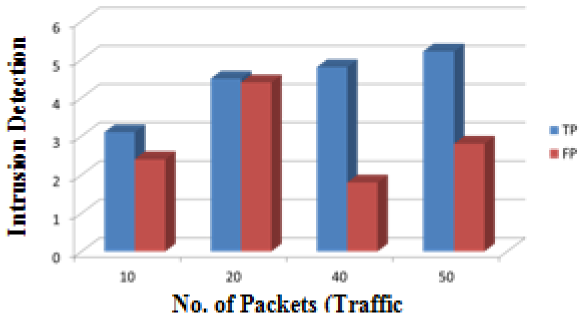

Figure 2: TP and FP detection rate vs no. of Packets in mirror-based approach

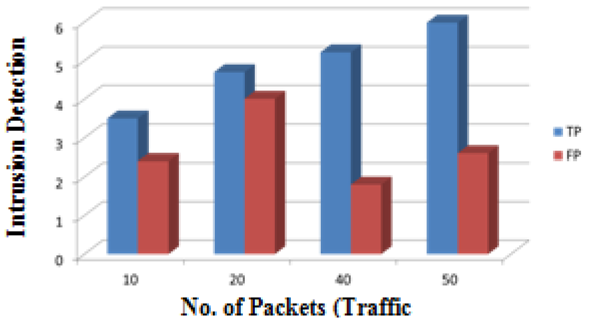

Figure 3: TP and FP detection rate vs no. of Packets in proxy-based approach

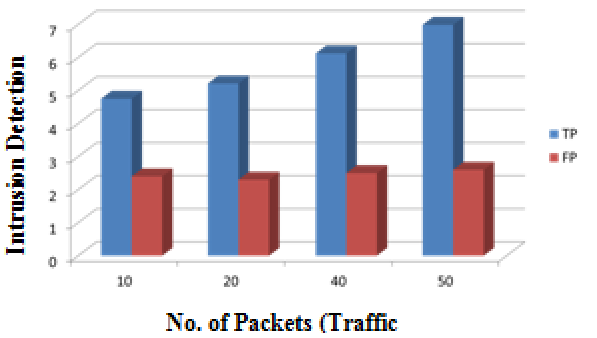

Figure 4: TP and FP detection rate vs no. of Packets in NICE

6. Conclusion

NICE is mainly implemented to detect and mitigate collaborative attacks in the cloud virtual networking operating environment. NICE uses the attack graph model to perform attack detection and prediction. By adding the concept of honeypot it tried to prevent the attacks before actually it happens. NICE has used KDD Cup 99 standard dataset of intrusions and developed its own KDD format through which it can read the data extracted from standard dataset files. To improve the detection accuracy and to cover the whole spectrum of IDS in the cloud system, NICE has incorporated host-based IDS solutions. For KDD extraction process and storing purpose NICE has used MySQL, which reduces the overall processing time for the dataset. The system performance evaluation demonstrates the feasibility of NICE and implemented solution can considerably moderate the risk of the cloud system from being exploited and misused by internal and external attackers.

7. Future Enhancement

As in the fastest growing IT world none of the system we can say is the 100 percent secure. Means every security system newly developed, one day definitely it will not be enough to fight with new security challenges and attacks. So, we can test implemented system with the help of more no. of security algorithms. NICE has used KDD Cup 99 Dataset which is inherited form DAPRA. So the implemented work can be extended with the direct use of DAPRA. But it's really a challenging job of handing such a huge amount of data. Also by incorporating concept of attaching mobile agent, it is possible to improve the intrusion detection probability and accuracy efficiently. Because of the use of mobile agent, black list present on one node can be easily be circulated to remaining nodes in the system. So that they will get intimated in advance before dealing with actual intrusion.

References

[1] Chun-Jen Chung, Pankaj Khatkar, Tianyi Xing, Jeongkeun Lee, Dijiang Huang, "NICE: Network Intrusion Detection and Countermeasure Selection in Virtual Network Systems", IEEE transactions on dependable and secure computing, vol. 10, no. 4, July/August 2013.

[2] Cloud Security Alliance, "Top Threats to Cloud Computing v1.0," https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf, Mar. 2010.

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

[4] Antonio Bianchi, Yan Shoshitaishvili, Christopher Kruegel, Giovanni Vigna, Blacksheep: Detecting Compromised Hosts in Homogeneous Crowds, CCS'12, October 16–18, 2012, Raleigh, North Carolina, USA.

[5] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee,BotHunter: Detecting Malware Infection through

IDS-driven Dialog Correlation,Proc. 16th USENIX Security Symp. (SS '07), pp. 12:1-12:16, Aug. 2007.

[6] G. Gu, J. Zhang, and W. Lee,BotSni_er: Detecting Botnet Command and Control Channels in Network Tra_c,Proc. 15th Ann. Network and Distributed System Security Symp. (NDSS '08), Feb. 2008.

[7] Oleg Sheyner, JeannetteWing, F.S. de Boer et al. (Eds.): FMCO 2003, LNCS 3188, pp. 344–371, 2004.

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

[9] NuSMV: A New Symbolic Model Checker,http://afrodite.itc. it:1024/nusmv. Aug. 2012.

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

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

[12] Kyle Ingols, Matthew Chu, Richard Lippmann, Seth Webster, Stephen Boyer, Modeling Modern Network Attacks and Countermeasures Using Attack Graphs

[13] S. Roschke, F. Cheng, and C. Meinel,A New Alert Correlation Algorithm Based on Attack Graph, Proc. Fourth Int'l Conf. Computational Intelligence in Security for Information Systems, pp. 58-67, 2011.

[14] L. Wang, A. Liu, and S. Jajodia, Using Attack Graphs for Correlating, Hypothesizing, and Predicting Intrusion Alerts, Computer Comm., vol. 29, no. 15, pp. 2917-2933, Sept. 2006.

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

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

[17] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani **“**A Detailed Analysis of the KDD CUP 99 Data Set”, Proceedings of the 2009 IEEE Symposium on Computational Intelligence in security and Defense Applications (CISDA 2009).

[18] Mitre Corporation, “Common Vulnerabilities and Exposures, CVE,” http://cve.mitre.org/, 2012.