Intrusion Detection using Snort based on NSG

[1]Ch. Sai Varshini, [2]D. Sai Praveen, [3]N. Veeranjaneyulu
Vignan's Foundation for Science, Technology and Research, Vadlamudi, Guntur, AP, India

Abstract: The increasing number of users and constantly evolving network technologies demands much more evolved cyber security mechanisms. Intruder agents like worms find their way easily through the World Wide Web and pose a great threat to the security systems. Defenders use poly trees to detect signatures based on their history. In this paper, we propose an intrusion detection system which works by automated signature generation system and signature-based method for intrusion detection.

IndexTerms **- Snort, Worms, Signatures, Common detection methodologies, IDS**

I. INTRODUCTION

Intrusion detection is an aspect of security measure which checks for network traffic that violates the security standards. Intrusion Detection System (IDS) is a type of software that detects the intruders if they do not follow security standards. This system can detect the attackers and inform this to the network administrator if it finds any disturbing flow of network in the system.

The cyberattacks detection methods have a method called signature based cyberattack detection which can be used as an intrusion detection system when they are initialized with a set of patterns or rules to identify an attack. The rules are usually compared or matched to the content of a network packet usually TCP/UDP header or payload. The intrusion detection system is used to increase the security levels of the computer network for a confined environment through detection for an IDS system.

There are many popular open source IDS software readily available on the World Wide Web. In this paper we use open source software called Snort. The advantage of using Snort is that it is open source software and can be easily modified with different set of rules or patterns from various sources that can be fed to the Snort engine.

The patterns that are used to detect the attacks and the rules that are defined are usually provided by experts from various cyber security communities. When the intrusion detection system faces deterministic attacks it is easy to create simple patterns that identifies the attacks and alerts the network. When faced with a malicious content like worms that uses same protocol to communicate in the network through command center or other third party software in the system. The main difficulty is observed when the system faces polymorphic worms and generates individual signatures every time the worm changes its code to represent as a new agent every time.

The Network based Signature Generation (NSG) algorithm is for identifying and assigning signatures to the polymorphic worms. In intrusion detection system we have two common types of IDS, Network Based IDS (NIDS) and Host Based IDS (HIDS) that are widely used.

Network based IDS (NIDS) attempts to identify anomalous and unauthorized based only on network traffic. The role of a network IDS is gathering, logging, identifying and informing. It regulates and verifies the traffic network looking for malicious activity or unauthorized activity.

A Host Based IDS (HIDS) examines all or few parts of the state of a computer system. It regulates and analyzes the internals like the network packets of a computer system. An HIDS gives you great visibility of what is happening on your important systems.

Both types of IDS technologies involve the collection and analysis of information from a wide variety of areas contained in a computer or a computer network to identify possible threats imposed by hackers from the outside of the organization or through the internal network. Still with these many advantages it is optimal for a organization to use both the Host based and Network based IDS.

One way the attack cannot be identified easily is when the attacker camouflages the worm with an encryption. In our case the many variations of the code can be generated when we encrypt the payload with various keys. The attacker has to pre hand developed a decryption technique that tries to catch the encrypted content.

Metamorphism, also called polymorphism, includes various techniques that can modify the code by itself. These methods use renaming of registers, the transposition of data in the code blocks, and the substitution of various instructions.

II. DETECTION METHODOLOGIES

IDS use the following methodologies for detecting content. Most IDS uses combination of multiple methodologies to detect threats.

1. **Signature Based:** This is the most commonly used method. This method is used in antivirus etc.,
2. **Anomaly-Based:** This method identifies the intrusions and classifies the events as normal or abnormal.
3. **Stateful -Protocol:** This method helps to identify the deviations of protocols in a state by comparing the events that are observed with predefined data which are accepted.

III. IDS TECHNOLOGIES

1. Signature-based intrusion detection technique:

This technique also referred to as knowledge based is based on searching for specific signatures that occur when something bad or when an attack happens. These types of techniques generate solutions which has fewer false positives than most of the other solutions. This is because the searching criteria are so fixed, and they also cover only signatures that are already in the database.

2. Anomaly-based intrusion detection technique:

This is also known as behavior-based technique. It defines various solutions, which is a difficult job, and sometimes leads to increase in false positive rate. Websites activity and cookies and few sites involving domains or URL's might be automatically blocked.

1. Network-Based: It usually deploys at the boundaries and monitors the flow of network.
2. Wireless: Monitors the wireless network traffic for malicious activity and it cannot verify the application or higher level protocols.
3. Network Behavior Analysis (NBA): Finds the threat that creates abnormal traffic flows.
4. Host Based: Monitors the events and characteristics of a single host or a single network.

IV. TOOLS

Snort: Snort is an open source network based intrusion detection and prevention system that has the ability to do real-time traffic analysis and perform packet logging on IP networks. It also performs protocol analysis, content matching and content searching. This program can be used to check for attacks, including semantic URL attacks, operating system fingerprinting attempts, server message block attacks, buffer overflows.

Snort can be made to work in mainly three modes:

1) Sniffer: In this mode, the program browses through the network packets and shows them on the console.

2) Packet Logger: In this mode, the program can log packets to the disk.

3) Network intrusion detection:In this mode, the application will monitor the network traffic and analyzes it against a rule set outlined by the user.

NMap: They provide variety of options for searching networks. These options measure protractile by scripts that offer additional vulnerability detection, advanced service detection and different options. Nmap will adapt to network conditions as well as latency and congestion throughout a scan.

Libpcap: Pcap denotes packet capturing. It consists of an API for capturing network packets. It is system independent software that works as a link between the network and the application. For systems that run on UNIX operating systems LibPcap is used. For systems that work on windows operating systems WinPcap is used.

The monitoring software (in this case Snort) uses LibPcap to capture packets that are travelling over a network.

V. PROPOSED METHOD

Intruders are a major threat to the network and its security aspects. They also reduce the performance of the systems. We use polytree and automatically improve the signature tree which is generated by the algorithm. By using this evolution technique the application will stay on par with new technologies and new signature models. It compares two signatures to determine if one is more specific form of the other or not.

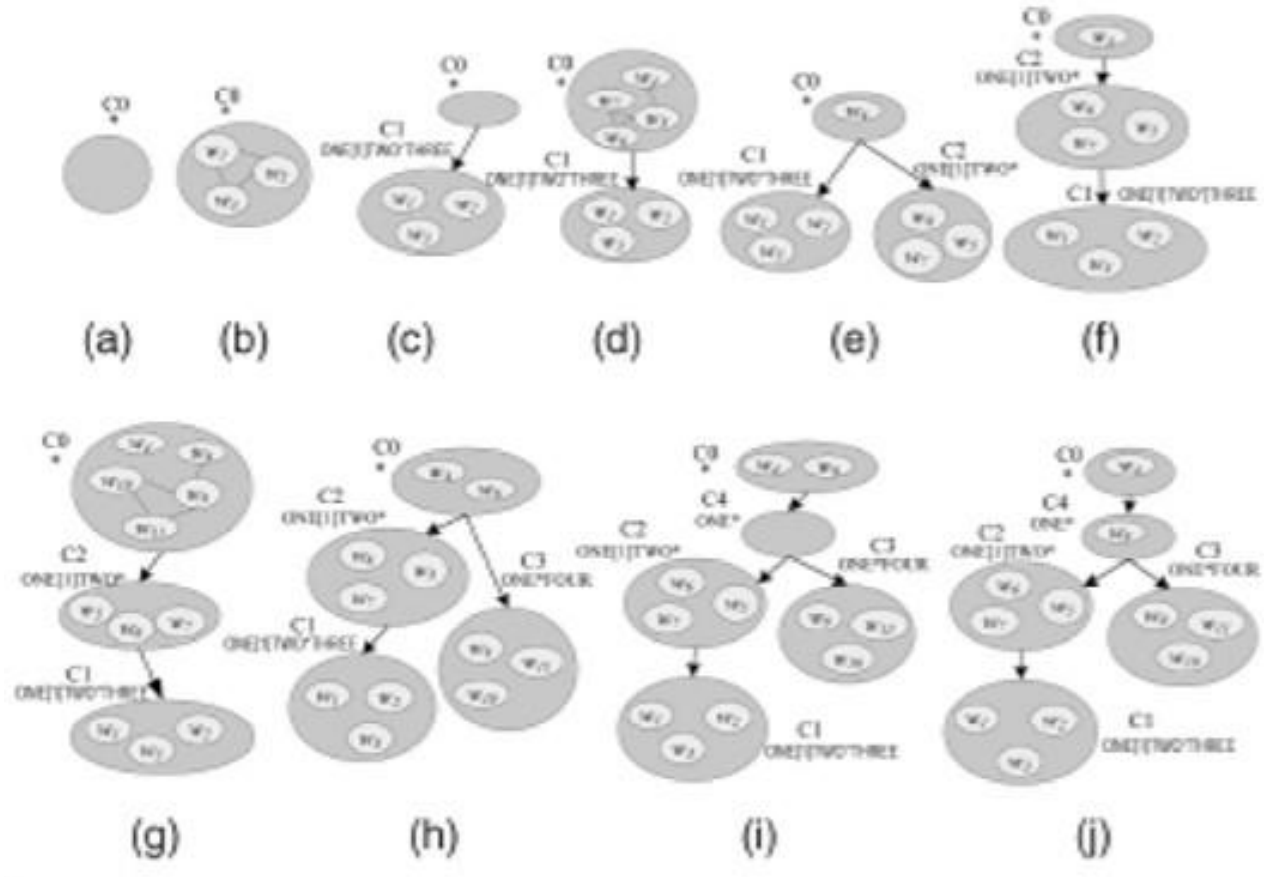

Figure 1: An example of signature tree which is automatically generated

Network Signature Tree Generation:

The signature tree generated from this algorithm is made up from signature generation method of the polytree system. This algorithm is divided into three steps with new sample j.

j-> new sample

s-> spliter

Samples Clustering: First we create or we take a environment where the worm sample have no noise with it. We have to use various samples that are generated from various worms and worm variants.

Refining Signatures:We create a signature tree let us call it is as *SigTree*. We take this tree as input and start pooling the traffic from the network to the tree. Now we get a set of signatures that are derived from the *SigTree.* The generated signatures are of low profile and have minimal suspicious traffic pool with protocols that are very limited.

Tree Construction:

We take *suspicious traffic pool M* , where *W*1, *W*2,…,*Wn*⊂M are the sample sets of worms *w*1, *w*2,…,*wn*, as input.

A tree with signatures as nodes such that every *wi* has a unique node in the *SigTree.*

Algorithm: IncSigTreeUpdate (j, s)

1. First we try to find a node let it be '*Pb*' in the signature tree. The tree should contain j belongs to '*Pb.sig*' and '*Pb*' is in the highest level of the tree. If there are several nodes which are similar to our choice we pick the node that is defined first and add it into this level.

//Step 1. Samples clustering

2. *Pb*← *Pb* U {j}

// Step 2:Signatures Refinement

3. When *Pb* has subset *Pnew* = {j1,j2,j3,⋯..,js} such that *Malign*(*Pnew*) < *Pb.sig*, then

// Step 3. Tree construction

Pb ← *Pb* / *Pnew*;

We add *Pnew* as a child of *Pb* to the signature tree;

// split a new node from *Pb* and name it as *Pnew*

Pnew.sig ← *Malign(Pnew)*;

SinkSamples(Pnew) ;

(i). for all node *P* such that *P* ≠ *Pnew*, *P:level = Pnew:level* and *P.sig* < *Pnew:sig*,

do MoveSubtree(*P*, *Pnew*); // make *P* as a child to *Pnew*

(ii) if *P* is a child of *Pb* and

Align(P:sig;Pnew:sig) < *Pb:sig*, then create new node *Pparent = {}* as a child of *Pb*;

Pparent:sig ←*Align(Pnew:sig;P:sig)*;

(iii) *MoveSubtree(Pnew, Pparent)* ;

MoveSubtree(P, Pparent);

// put *Pnew* as a child of *Pparent*

(iv). *SinkSamples(Pparent);*

SinkSamples: A sample j is moved from *P* to *Pt* if

Pt.level = P.level +1 and *j< Pt:sig*

Move Subtree: In the tree, the subtree with root Ps becomes a child to the tree with Pt as root.

Generating Signature tree for polymorphic worms:

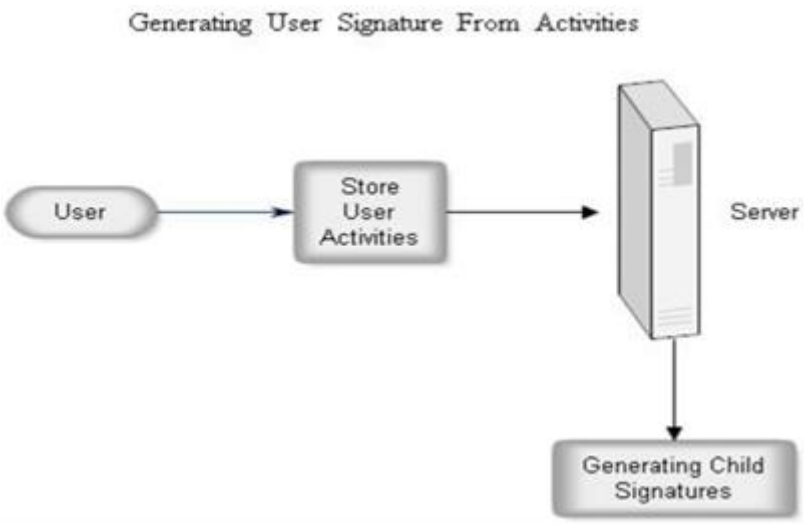

Figure 2: Generating user signature from activities

To create signatures for attacks by using the different parts and any two signatures can be used to compare in terms of specifics based on the definitions. The tree wills visual logical relation between the attacks and how worms evolve.

VI. CONCLUSION

The protection against automated intrusion agents is very weak when compared to other types of security standards. So it is better to detect and eliminate these types of intrusion agents as quickly as possible. The use of an intrusion detection system along with a network signature tree will help in automated process of protection. The future enhancement will be using this idea in industrial sectors where the confidentiality is at big risk.

REFERENCES

[1] B. Karp, J. Newsome, D. Song, "Polygraph: Automatically Generating Signatures for Polymorphic Worms", Proc. 2005 IEEE Symp. Security and Privacy, pp. 226-241, 2005.

[2] Rafeeq Ur Rehman, Intrusion Detection Systems with Snort: Advanced IDS Techniques with Snort, Apache, Prentice Hall PTR

[3] Yong Tang, Bin Xiao and Xicheng Lu,"Signature Tree Generation for Polymorphic Worms", Proc., IEEE Transactions on, pp. 565 - 579, 2011.

[4]Y. Song, M.E. Locasto, A. Stavrou, A.D. Keromytis, S.J. Stolfo,"On the Infeasibility of Modeling Polymorphic Shellcode", Proc. ACM Conf. Computer and Comm. Security (CCS), 2007

[5] SNORT. Project homepage, http://www.snort.org/

[6] Choraś, M., Kozik, R., Puchalski, D., Hołubowicz, W.: Correlation Approach for SQL Injection Attacks Detection. In: Herrero, Á., et al. (eds.) Int. Joint Conf. CISIS'12-ICEUTE'12-SOCO'12. AISC, vol. 189, pp. 177–185. Springer, Heidelberg (2013)