RUDRAA – an int**RU**sion **D**etection & p**R**evention sign**A**ture formul**A**tion process

N. Subramanian
Centre for Development of Advanced Computing (C-DAC), #68, E-City, Bangalore, India – 560 100
subbu@ncb.ernet.in

Sachin Narayanan
Centre for Development of Advanced Computing (C-DAC), #68, E-City, Bangalore, India – 560 100
sachin@ncb.ernet.in

Mohammed Misbahuddin
Centre for Development of Advanced Computing (C-DAC), #68, E-City, Bangalore, India – 560 100
misbah@ncb.ernet.in

Bishwa Ranjan Ghosh
Centre for Development of Advanced Computing (C-DAC), #68, E-City, Bangalore, India – 560 100
ghosh@ncb.ernet.in

ABSTRACT
Intrusion Detection & Prevention Systems generally aims at detecting / preventing attacks against computer systems and networks or in general against information systems. The basic task of IDPS is to monitor the usage of such systems by detecting / preventing any unwarranted incidents which leads the systems to insecure state. The monitoring is done by checking each packet for its validity against the signatures formulated for identified vulnerabilities. Since, signatures are the heart & soul of an Intrusion Detection and Prevention System (IDPS), we, in this paper, discuss the methodology we adapted in our research effort in the domain of Intrusion Detection and Prevention (IDP) called RUDRAA for formulating, verifying & validating the potential signatures to be used with IDPS. The research objectives of this project are 1) To formulate potential IPS signatures to be used with Intrusion prevention system. 2) To do code improvement and packaging of IDP software solution.

Categories and Subject Descriptors
K.6.5 [**Management of Computing and Information Systems**]: Security and Protection – *Authentication, Invasive software, unauthorized access and Verification*

General Terms
Design, Reliability, Security, Verification.

Keywords
Intrusion Detection / Prevention Systems, Signature Formulation, Vulnerability, Bug Tracking,

ICAC3'09, January 23–24, 2009, Mumbai, Maharashtra, India.

1. INTRODUCTION

1.1 Domain Background
An Intrusion Detection & Prevention System (IDPS) is software and/or hardware designed to detect / prevent unwanted attempts at accessing, manipulating, and/or disabling of computer systems, mainly through a network, such as the Internet. These attempts may take the form of attacks, as examples, by crackers, malware and/or disgruntled employees. An IDPS[5,6] cannot directly detect / prevent attacks within properly encrypted traffic.

An IDPS is used to detect / prevent several types of malicious behaviors that can compromise the security and trust of a computer system. This includes network attacks against vulnerable services, data driven attacks on applications, host based attacks such as privilege escalation, unauthorized logins and access to sensitive files, and malware (viruses, Trojan horses, and worms).

An IDPS can be composed of several components: Sensors which generate security events, a Console to monitor events and alerts and control the sensors, and a central Engine that records events logged by the sensors in a database and uses a system of rules to generate alerts from security events received. There are several ways to categorize an IDPS depending on the type and location of the sensors and the methodology used by the engine to generate alerts. In many simple IDPS implementations all three components are combined in a single device or appliance

All Intrusion Detection Systems use one of two detection techniques: statistical anomaly based and/or signature based.

Statistical anomaly based IDS- A statistical anomaly based IDS establishes a performance baseline based on normal network traffic evaluations. It will then sample current network traffic activity to this baseline in order to detect whether or not it is within baseline parameters. If the sampled traffic is outside baseline parameters an alarm will be triggered.

Signature based IDS- Network traffic is examined for preconfigured and predetermined attack patterns known as signatures. Signatures have attributes like false-positives, false-negatives, completeness, breadth, precision, collision and recall

[9]. Many attacks today have distinct signatures for vulnerabilities. In good security practice, a collection of these signatures must be constantly updated to mitigate emerging threats.

Attack analysis is the process of conducting systematic analysis on multi-stage attacks and facilitating large scale detection and visualization of security events by embedding modeling and analytical components within a more expansive security framework and basically deals with Penetration testing, Ethical hacking, Incident handling and Exploit writing.

1.2 The Need

Current weaknesses in IDP's include the fact that new attacks are not detected until someone has generated a rule or signature that picks up that specific attack and that most attacks need only be slightly altered in order to bypass existing rules. New signatures are generally created manually by addressing the application vulnerabilities. Some signatures may tell you which specific attack is occurring or what vulnerability the attacker is trying to exploit, while other signatures may just indicate that unusual behavior is occurring, without specifying a particular attack. So, it is essential to craft different types of signatures to identify the potential attack. To validate the IDP signatures, it's necessary to carry out the validation and attack analysis.

Bug fixing mechanism will enable the end user to announce bugs found in the IDP software, and, further developer refers the bug details to fix it. End user may check the status of announced bug and, if it has fixed, then, they can update their existing version of IDP with new release.

1.3 Purpose & Aim

We have already built an IDS and we are currently developing a hardware based IPS for which we need to have our own proprietary signatures, since the current signatures which are being used by us in IDS are from public domain and we need to verify and validated these signatures for using with IPS. Moreover, in order to provide an improved and customized IPS, we need to perform attacks against various exploits in order to identify its impact on the network and the victim system.

1.4 Objectives

The major objectives of this project are as follows:
1) IDP (Intrusion Detection Prevention) signature formulation.
2) Bug Tracking, Code improvement and packaging of IDP

software solution.

This project is intended to formulate IDP signatures and provide a mechanism for code improvement and packaging for IDPS. Since signatures are the soul of an IDPS, it becomes essential to accurately formulate the signatures. A lot of signatures are available in public domains which are not verified/validated for an IPS. So, in order to do so, the process of formulating the signatures is divided into three phases.

- Signature Identification
- Signature Verification
- Signature Validation and Impact Analysis

Signature identification involves identification of potential IPS candidate signatures targeting applications/class based on their relevance and severity which have publicly available references like CVE[1,4], NESSUS[3], BUGTRAQ[7] and others.

Signature verification is the phase where the formulated IPS Signatures are evaluated for the appropriate fields related to headers, protocols, payload, state based, and action types. This involves header analysis, payload analysis, state-based analysis and action type analysis.

Signature validation and impact analysis This includes the Attack analysis for all IPS signatures. To perform this crucial activity, setup of separate attack analysis lab is essential. Attack analysis lab needs to be equipped with all widely used open source attack/exploits. Since Signature formulation is a continuous activity, the above mentioned three phases would continue in future also.

Code improvement and packaging involves fixing of software bugs, addressing bug-tracking issues while improving the code, release management and packaging for BOSS, Kubuntu, Debian, RHEL, CentOS and FC for various kernel versions.

The outcome of this activity is a set of formulated IDP signatures and a mechanism for addressing bug-tracking issues, code improvement issues and packaging.

1.5 Alternatives

Almost all IDP solution providers do have the signatures formulation mechanism in place, such as, Cisco, iPolicy, real secure, etc... Even some open source projects exist which formulate signatures on regular basis. All of these agencies do not publicize their IPS signatures. SNORT IDS signatures are publicly available, but all of these IDS signatures can not feed into the IPS system without verification and validation.

2. Approach / Methodology

To achieve the objectives of this project, we approached the problem statement categorically in the following way:

Signature formulation: This process comprises of three phases, I) Signature identification II) Signature verification and III) Impact Analysis. To do this we took the Snort's 2.8.1 signature set which counted to 13300+ IDS signatures. The signature formulation process is depicted in Fig 1.

I) Signature Identification- In this phase we filtered the 13300+ Snort signatures [8] based on Application Relevance, Severity [1,4] (signatures having severity greater than 6) and Time (signatures notified after year 2000). This is shown in figure 1 as the output of S1 & S2.

For Application/Class Relevance, we filtered the potential IPS candidate signatures targeting applications/class based on high(80%) and medium(20%) severity relating to HTTP, DNS, SMTP, SQL, RPC, POP3, Telnet, R-Services, NetBIOS, Finger, Chat, Backdoor, Attack Response and Exploit based vulnerabilities.. These potential candidate signatures will have references to one of the following CVE, NESSUS, BUGTRAQ, OTHERS {all high severe signatures do not have the CVE

reference}. The resultant set after filtering according to S1 contained 5480 IDP signatures which were again subjected to filtering based on severity (process S2) and yielded 1407 candidate signatures. So, at the end of identification phase we had 1407 candidate IPS signatures.

11) Signature Verification-
The signature verification and validation is done by doing the following four analysis:

Header analysis:
This includes verification of the protocol fields value under which a signature falls, such as ARP, ICMP, IP, TCP, UDP etc...

Information about the IP (external/home network). This field has to be properly verified because the IP field value reveals the information about direction of packet transmission. IP field can have different values like, EXTERNAL_NETWORK, HOME_NETWORK, IP_RANGE and any.

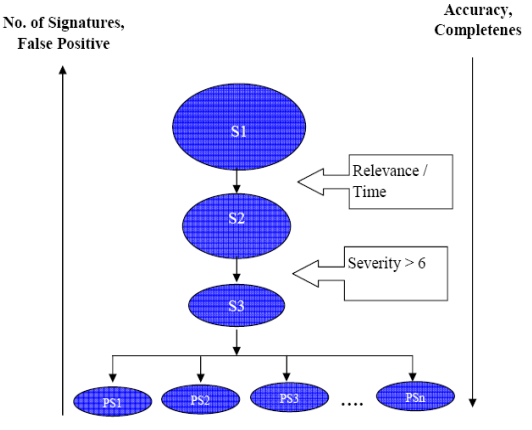
Port field in signature specifies about the destined application for which the signature has formulated. This field may be assigned with different value like HTTP_PORT, SMTP_PORT, 80, 21, range (40001:56000) and any.

Other protocol related fields like flags, Type of service, IP Identification number, various sequence and acknowledgment number, TCP window size, icode, itype, icmpid etc. has to be verified for signature.

Payload analysis: Most of the signatures have content (payload) or value like 'uricontent' which can be either in the plain text or in regular expression form. To carry out Deep packet and content analysis, the content and uricontent fields are supported by other fields such as offset, depth, byte jump, meta data, case or nocase, distance, within, etc

Sate based analysis: This includes the verification of fields like, Flow information, connection state, and session information.

Fig 1: Signature Formulation Process



S1: IDPS Signatures from SNORT
S2: Possible IPS signatures
S3: Potential IPS Signatures
PS: Potential IPS Signatures Classified on basis of application type

Action type analysis: After successful matching of each signature, a specific action has to be associated with it. Like DROP, LOG, ACCEPT. So, based on the severity and impact of attack pattern (signature) the action type value has to be properly assigned and verified further. Based on the action type analysis we classified the 1407 signatures as 'IPS DROP' which counted to 655 and 'IDS ALERT' which counted to 752 signatures respectively.

III) Impact Analysis- This includes the Attack analysis for all candidate IPS signatures which is the output of signature verification phase.

To perform this crucial activity, we have setup a separate attack analysis lab. The architecture of this attack lab is shown in fig 2. We have categorized the systems as 12 Victim systems, one IDP sensor running on high end server with dual interface card. We also used 'n' (=20) attacker systems for performing various attacks based on the type of vulnerability. All these systems are connected with two switches respectively. All the systems in the attack lab network were configured with various operating systems such as Windows XP, 2000, Fedora 9, Red Hat Linux, Debian etc. These systems are installed with all widely used open source attack/exploits softwares. The different tools that were used to perform the Impact analysis are: Nemesis, Nmap, Nessus, Metasploit Framework, Wireshark, Tcpdump, Netcat, Nikto, Paros proxy, Whisker, IDSwakeup, etc…

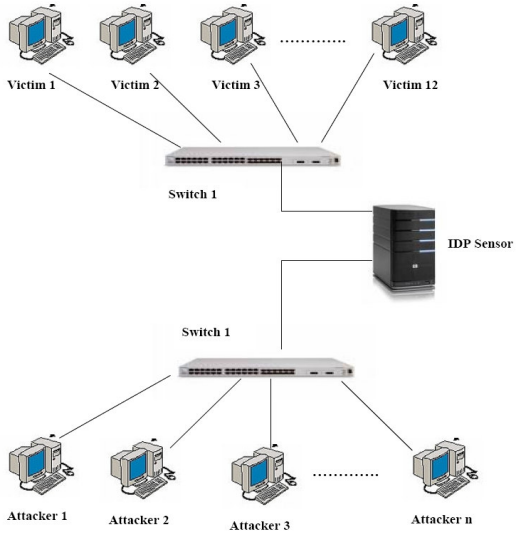


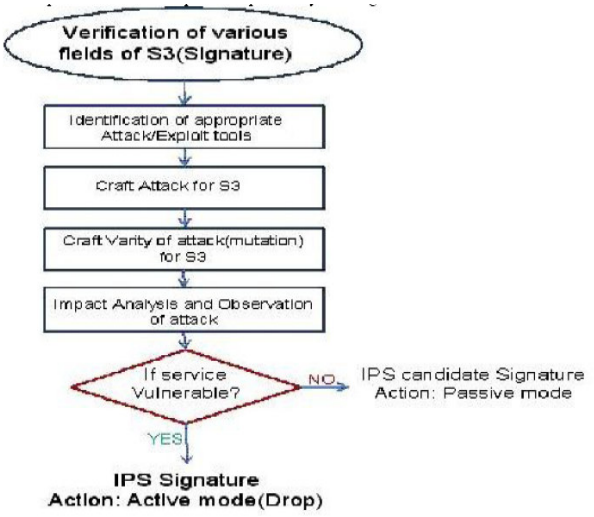
Fig.2 Architecture of Attack Lab

The activities involved in Impact analysis includes:

1) Identification of Vulnerability in the system as specified by the formulated IPS signatures.
2) Exploiting the vulnerability by using scripts, codes or exploits available in the open domain
3) Observation and analysis of the impact of the crafted attack using various walk scenarios.

4) Decision on the action relevant for the caused impact.

In addition, the packet captures and the impact analysis of each attack is recorded for input to the IPS project. The detailed



Impact Analysis activities are shown in Fig 3.

Fig 3: Impact Analysis

Bug-tracking, Code improvement and Packaging

This phase involves:

1) Packaging for BOSS, Kubuntu, Debian, RHEL, CentOS and
1. FC for various kernel versions.
2) Release management- release notes and installation manuals.
3) Devising a bug tracking mechanism and addressing bug
2. tracking issues
4) Devising a mechanism for classifying configuration and code
3. related bugs.
5) Fixing of software bugs.
6) Code improvement

3. ACKNOWLEDGMENTS

Our special thanks to Rajiv Ranjan who has put all his efforts to initiate this research work and giving it a shape.

4. REFERENCES

[1] http://cve.mitre.org/

[2] http://en.wikipedia.org/wiki/Intrusion-detection_system

[3] www.nessus.org

[4] http://nvd.nist.gov/

[5] http://www.sans.org/reading_room/whitepapers/detection/

[6] http://www.sans.org/reading_room/whitepapers/intrusion/

[7] www.securityfocus.com

[8] www.snort.org/

[9] Writing Detection Signatures – Christopher Jordan www.usenix.org/publications/login/2005-12/pdfs/jordan.pdf