EMIST Network Intrusion Detection (NID)

Tool Manual (Version I)

J. Wang, D.J. Miller and G. Kesidis

CSE & EE Depts, Penn State

TABLE OF CONTENTS

1. Overview

The EMIST NID (Network Intrusion Detection) tool is an off-line network intrusion system capable of detecting suspicious network traffic and extracting their corresponding signatures. The EMIST NID tool performs 5D multidimensional header-based clustering and anomaly detection to identify suspicious traffic, from which it can automatically extract suspicious signatures by applying a Generalize Suffix Tree based method.

2. Getting Started

2.1. System Requirements

The EMIST NID tool can run on Window XP, Linux Fedora and UNIX Solaris, and it is recommended to run the tool with 1.0 GHz or higher CPU and 256MB or higher memory.

The EMIST NID tool may need 50 to 70MB memory, and its processing time depends on not only the length but also the traffic characteristics of input trace. Generally speaking, the more the involved worm traffic, the longer the processing time. It is recommended the input trace be smaller than 1GB. It takes 15 seconds for the EMIST NID tool to process the worm-salted trace provided in our example with Intel 3.0 GHz CPU and 1GB memory. The running time could be much longer, even many minute, for some worm traces. So please be patient when running the tool.

2.2. Installation

Simply download the executable file for Windows XP, or the source code (including makefile) for Linux/UNIX from http://emist.ist.psu.edu/download_IDS.html.

3. Input Parameters

The EMIST NID tool version I only supports the standard TCPDUMP format (Appendix I). Users only need to specify the path of input TCPDUMP file.

4. Output Report

The detection report of the EMIST NID tool will be saved automatically in the same folder as the input trace file with the name 'Report.txt'. The report includes two parts: basic information of input trace, and suspicious network traffic and their corresponding signatures.

4.1. Basic Information

This part includes the basic information of the input trace: starting time (EST time), time interval, total bytes and number of packets in the traffic, and also the threshold used for multidimensional clustering.

4.2. Suspicious Traffic and Signatures

In this part, suspicious network traffic is identified into clusters based on packet header five-tuple: source IP, destination IP, source port, destination port, and protocol, together with their corresponding bytes, packet numbers and percentage of total traffic.

For each suspicious cluster, its signatures are extracted from the payloads of all packets which satisfy the cluster definition. Here, "Length" stands for the length of a signature, and "Freq" stands for the times it occurs in the suspicious cluster. Signature binaries are also recorded in the report.

Notation Description:

*: This dimension is not used in defining the flow, i.e., the flow includes all of the possible values in this dimension;

Slash (/) in IP dimensions: network mask of IP addresses

'low port', 'high port' in port dimensions: 'low port' means the port group of port number less than 1024; 'high port' means the port group of port number larger than 1023.

5. Contact

Should you have any question or suggestion to the EMIST NID tool, please contact Jisheng Wang at jzw128@psu.edu.

Thanks for using EMIST NID tool!

References

[1] J. Wang, D.J. Miller, and G. Kesidis. “Efficient Mining of the Multidimensional Traffic Cluster Hierarchy for Digesting, Visualization, and Anomaly Identification”, IEEE JSAC on High-Speed Network Security, 2006.

[2] J. Wang, I. Hamadeh, G. Kesidis, and D.J. Miller. “Polymorphic Worm Detection and Defense: System Design, Experimental Methodology, and Data Resources”, SIGCOMM Workshop on LSAD, 2006.

Appendix I: Standard TCPDUMP Datagram Format

The standard TCPDUMP datagram is defined as following (Please refer to http://www.tcpdump.org/ for details):

```
struct tcpdump_file_header{
      u_int32 magic;    /* either 0xa1b2c3d4 or 0xa1b2cd34 */
      u_int16 major_version;
      u_int16 minor_version;
      int thiszone;    /* GMT to local correction */
      u_int32 sigfigs;    /* accuracy of timestamps */
      u_int32 snaplen;    /* maximum length saved portion of each packet */
      u_int32 linktype;    /* data link type */
}
```

Each packet will have the following information:

```
struct pkt_header{
      u_int32 seconds;
      u_int32 microseconds;
      u_int32 caplen;    /* length of portion present */
      u_int32 len;    /* length of the packet (off wire) */
}

struct ethernet_pkt{
      u_int16 usrc;    /* upper 2 bytes of the source address */
      u_int32 lsrc;    /* lower 4 bytes of the source address */
      u_int16 udst;    /* upper 2 bytes of the destination address */
      u_int32 ldst;    /* lower 4 bytes of the destination address */
      u_int16 mode;    /* 0x800 for IPv4 */
}

struct IP_headher;
struct TCP or UPD header;
```