A Survey on Methodologies for IDS Testing

Marmagna Desai *

March 17, 2004

Abstract

The development of Intrusion Detection Systems is the subject developed in recent years. The concepts for building an IDS were inherited from many different fields like ACL (Access Control Lists), Virus Detection Tools, Firewalls and security features in UNIX. Though these concepts are fairly developed and well tested the approaches taken by IDS are not mature. The greatest difficulty for this development is inability to create generalize methodology or test-bed environment for testing IDS before deploying it in the real production world. This survey is conducted by taking two Papers into consideration. The one is written in October 1996, describing one methodology for testing IDS. This methodology is based on the concepts of Software Engineering Testing adapted for testing IDS. User-simulation Scripts and UNIX tool "expect" are some of the techniques explored by this early effort to develop a methodology for IDS testing. The second paper is written in 2003. This work can be viewed as conclusive comment for almost all methodologies thought in the past for IDS testing. It discusses almost all major development in this field including DARPA and LARIAT environments, tools like IDSWakeup, Flame Thrower, WebAvalanche, TCPReplay, FragRouter and HPING2. The discussion includes line of thought, achievements and shortcomings of each effort. Before this paper conclude the inadequacy of the Open source evaluation environment and methodology for researchers and developers to test and develop new IDS, it discusses five different major evaluation environments for IDS testing. This paper is casting light on the various needs for such open source environment and clears the sight for such developments.

1 Introduction

IDS attempts to identify unauthorized use , misuse, and abuse of computer systems[1]. In order to develop such a system it is vital to test the IDS on various platforms and in various conditions. Recent IDSs are developed to work with Windows, UNIX and Network environments. The testing of such system includes variety of attacks, different load conditions on Host as well as Network, various audit trails and process system-calls, analysis of such logs in meaningful way and the most importantly the real-time traffic generation to depict actual traffic on network. Though these issues are clear to developer community, it is rare to see that all are implemented in any one testing-bed so far. The first section of this survey discusses the early effort made by [1] in October, 1996. The second part of this survey is about the second paper referred which is developed in 2003 as the commentary on such efforts to develop testing environments in the past. Finally this survey concludes with some of the good references and bibliography.

*©2004 UoW. Marmagna Desai [101282813]

2 A Methodology for Testing IDS - Paper I

This methodology consists of techniques from the field of software testing which were adapted for the specific purpose of testing IDSs. Initially the paper discusses general IDS performance objectives which are the bases for this methodology. It also quantitative results from testing experiments and overview of the software platform that they have used to create user-simulation scripts for testing. The effort also expand UNIX tool *expect* which enables concurrent scripting and record-and-play features for simulating real-time environment.

2.1 Introduction

Testing IDS involves impossibility to identify the set of all possible intrusions that might occur at the site where a particular IDS is installed and difficulties in evaluating an IDS where it can be affected by various condistions in the computer systems. This paper develops methodology which is based on a set of general IDS performance objectives, such as the ability to detect a broad range of known intrustions. It consists of stategies for selecting test cases, and a series of detailed testing procedures. The concepts of *Software Engineering* and UNIX tool *expect* as platform for creating user-simulation scripts are used extensively in this effort.

2.2 Motivation

Based on the fact of concurrent and distributed intrusions on victim network/hosts, this paper classify intrusions in following categories:

- Single Intruder Single Terminal: Intrusions are launched by single intruder from single terminal or logical equipment.
- Single Intruder Multiple Terminal: This category consists of single intruder establishing many connections to different victims from single terminal using various session-windows.
- Multiple Intruder Multiple Terminal: Here the intrusion is more in distributed way where multiple users participate in one or many intrusions from multiple terminals.

There are two major approaches used by IDSs to detect intrusive behavior. Anomaly Detection and Misuse Detection. The prior is based on the premise that an attack on the computer system/network will be noticeably different from normal activity. Hence it models normal behaviors of the users and compares abnormality with behavior models. It also sets thresholds for anomalous occurrence beyond which the activity will be identified as intrusion. In misuse-detection method, IDS watches for indications of specific, precisely representable techniques for computer system abuse[1]. IDS develops *signatures* which are encapsulations of well-knows attacks and intrusion techniques. Though both approaches can be deployed at computer system or network boundary, intrusions are *not*likely to be detected by analysis of the audit records of any single computer in teh network. Hence this paper uses term *network intrusion* to refer to intrusion that involves more than one computer in a network.

2.3 Software Platform

This methodology simulates computer users - both intruders as well as normal users- while the IDS is running. The UNIX package *expect* is used for this purpose which is based on TCL (Tool command language). *Expect* provides way to write scripts that include intrusive commands. TCL component provides an interpreter for a simple programming language that includes variables, procedures, control constructs such as "if" and "for" statements, arithmetic expressions, lists, strings, and other features. *TCL* is implemented in C library package. To accommodate "reproducible testing" or "replay" *[similar to tcpreplay]* the authors have developed a synchronization mechanism. This mechanism provides a means for the programmer to establish a fixed order of execution for key events.

2.4 Testing Issues

In order to proceed further to describe methodology for testing IDS, authors of this paper identify three primary design goals for test-bed.

- **Broad Detection Range:** The testing should be performed based on the broad range of attacks and vulnerabilities in order to identify intrusions from normal behavior for an IDS.
- **Economy in Resource Usage:** The testing environment should not CPU or Memory consuming activity on a system where the testing itself becomes problem for normal functionalities.
- **Resilience to Stress:** IDS should work in high load and stressful environments. Hence testing should be designed for such situations also.

Before defining exact methodology the authors are selecting test-case for their experimentation. For IDS testing it is vital to simulate "intrusions" and "normal behaviors". Next problem is which intrusions to simulate for test. To deal with this problem authors are proposing to classify a huge database of attacks collected by CERT advisories, COPS, 2600 or PHRACK kind of resources. Though this classification is the crux of the problem in building environment for testing IDS because it is impossible to define perfect class which includes all attacks. Novel attacks are still not detectable or included in such approach.

2.5 Testing Methodology

Authors are proposing a set of detailed procedures for testing IDSs. These procedures can be applied to Host and Network based IDS. The most general procedure steps for any variance of testing scenarios is as:

- Create and/or select a set of test scripts.
- Establish the desired conditions (*background activity*) in the computing environment.
- Start the IDS.
- Execute Test Scripts which simulates user behaviors.
- Analyze the IDS's output.

In their work the authors are testing IDS in basically three different kind of environments. Each of them have different goals to be achieved. Namely, they are, **Intrusion Identification Test**,**Resource Usage Test**, and **Stress Test**.

2.6 Intrusion Identification Test

This test indicates how well the IDS detects intrusions. Hence basically it reflects an ability of an IDS to distinguish intrusions from normal behavior. This test also included Normal User test, which measured how well IDS identifies Normal behavior and hence it reflects the amount of false positives generated by IDS. The methodology for this test is exactly as the general methodology described in earlier section.

2.7 Resource Usage Test

It measures how much system resources are used by the IDS. The significant of this test is, it reflects the appropriateness of particular IDS to be used in given computational environment.

[Example: Disk Space Test -

- *Eleminate unrelated activiy in the test environment.*
- *Start the IDS.*
- *Run the test script for a measured period of time.*
- *Calculate the total disk space used by the IDS to record the session associated with the script.*

2.8 Stress Test

Under stressful conditions IDS may not detect certain attacks and hence testing particular IDS in such environment is must for complete testing. Authors have developed different kind of stresses to produce possible situations. And also suggested methodology for each of such stresses.

- Smokescreen Noise: Authors define *noise* to be computer activity which is not directly intrusion activity. Hence it can be produced by intruder to disguise the attack. By indicating the steps to test authors propose to create different set of scripts which will be treated as supplement to main attack scripts and act as *noise*. The output of such test will be compared with the simple intrusion detection test's result. And hence the comparison will show the side-effect of noise introduced in the system.
- Background Noise: The only difference between prior and this is it is created by legitimate users. And hence not intended to disguise any attack. Same idea is used to compare the two results of tests and decide on the IDS performance.
- High-Volume Sessions: The volume test checks how the IDS is affected by high-volume sessions. The authors also define high-volume which is dependant on the nature of IDS. The purpose of this test is to check if the IDS monitors the high-volume sessions and other sessions correctly.

There are also other tests which are performed under different Load test and Intensity conditions. The result of the implementation of the methodology can be summarized as given in the next section.

2.9 Experimental Results

The experiments/tests were performed on Network Security Monitor developed at University of California, Davis. NSM can monitor all packet traffic and associate each such packet with the corresponding computer-to-computer connection[1]. The following subsections provides results on such software for each tests:

- Basic Detection Test: *Expect* scripts were used to simulate a specific intrusive command sequence. The behaviors like browsing a directory, password cracking, password guessing using dictionary, attempt to modify system files, excessive network mobility and exploiting super-user access vulnerability were simulated by the UNIX tool. Though NSM produced low value warnings for connections associated with intrusive scripts, authors believe that like any IDS, NSM can be tuned to be more accurate in detection.

- Stress Tests: The authors have hypothesis that stress in the form of high load can affect the NSM's ability to monitor network connection. Hence, they have performed the Load Stress Test for NSM which monitor TCP packets with *telnet* connection. The load scripts where designed to create various levels of loads and comparing NSM's result for each tests. The reports were **affected** by the loads, where ideally they should be identical. Apparently higher load tests missed some network packets. In general tests results indicate that an IDS can be affected by stressful conditions, and it may be potentially vulnerable to an attack by an intruder who knows how to exploit this weakness[1].

2.10 Conclusion

This whole project can be seen as one of the many implementations of Testing Environment design approach. The major limitation of this effort is the software platform that authors using can not completely simulate the user behavior working with a GUI based environment like X windows or windows. However authors don't agree to the necessity to simulate complete behavior of user. The second limitation is they have used only misuse detection approach to simulation, which will not work for novel attacks. Though they include this task in future extension of their work. Apart from this they have concluded it as an useful effort to know more information about IDS and its capability. A good testing methodology to start with.

3 Comments on Paper I

The paper discussed in prior part of this survey is an experiment done in 1996. Since then there are many efforts to make testing environment better and more general. Though they are not successful yet, they provide great amount of knowledge in this new field. The following paper which is the recent survey of almost all such approaches is summarized in my own words.

4 Intrusion Detection Testing and Benchmarking Methodologies - Paper II

This paper is a survey on existing methodologies for IDS testing and evaluation environment development. The authors study and analyze carefully the past efforts of build such environment including non-open source, government R & D projects DARPA and LARIAT. Such tools also include NIDSBench and IDSWakeup, FlameThrower, WebAvalanche, TCPreplay and HPING2. The second part of this survey discuss issues regarding generating realistic evaluation environment. Authors also discuss examples of different examples of such environments like creating custom software, advanced security audit trails analysis and vendor independent testing lab. Though these efforts are great deal of work in themselves do not creates true generalized environment for IDS evaluation. Authors conclude by stating "We do not believe that there is ever going to be a comprehensive test nor a single methodology that would examine an IDS for every attack, verify its abilities and identify all its weakness."[2] However they believe that it is possible to improve and create an open source frame work for testing IDS.

4.1 Existing Tools and Methodology

- **DARPA Environment:** This effort was made in 1998 by DARPA which was first systematic effort to test IDS. The general approach to testing was to execute an off-line and on-line evaluations. In off-line evaluation, a set of seven weeks of training data was provided to vendors. This consisted of sessions, which were marked as normal, or as attacks. The off-line training data was intended to use as a tool for the IDS experts to tune and optimize their system. A second set of two weeks of data was generated which was to be used for actual testing. The tool *tcpreplay* was used to feed the traffic to the evaluation system.[2] The major issue was how to generate traffic. There were three options. First was to, collect real operational data and attack an actual organization. The next options was to actually go ahead and sanitize the data collected and then introduce the attacks in the data. Both of them were not practical and unacceptable. Hence designers of DARPA project used third approach. That was to synthesize all sessions from scratch. This will generate non-sensitive traffic similar to operational traffic. Many of the attacks were analyzed beforehand to verify that they function in the test bed. At the end of this experiment following suggestions were given in 1999:

– Victim Windows NT machines should be introduced. Hence the experiment was on only UNIX based systems.
– More new stealthy attacks were added to avoid IDS detection.
– Analysis of misses and high-scoring false alarms should determine the cause of detection misses and false alarms.
– Participants were allowed to submit information aiding in the identification of many attacks and their appropriate response.
– Detection of Novel attacks without first training should be considered.

Though DARPA project considered and worked on these issues latter for some time, the final implementation was extensively criticized. Following were the major among them:

– It was unknown that how many false alarms, would background traffic alone generate.
– Neither the statistics of the real traffic nor the statistics that the generated traffic was supposed to match were ever published.
– Data rates and their variations with time was never a variable in the DARPA tests.
– The attacks were evenly distributed throughout the background traffic where in reality it will not be.
– There was no control group employed in the testing.
– There were inconsistencies in documentation presented by DARPA.

In spite of these criticism, the efforts were significant contribution to the goal. And the data gathered by this effort is still used by other developers in 2002.

- **LARIAT Environment:** This project was recently developed as a part of the development of comprehensive testing environment for IDS, firewalls and ACLs. LARIAT emulates the network traffic from a small organization connected to the Internet[2]. The user selects the type of the attack and traffic to be generated in background. The system verifies that network is ready before the test begins. Then, it initializes the network and configures the hosts. During the next strep the test conditions are set up which includes generation of traffic and attack scripts, starting logging services and scheduling the attack. LARIAT allows user to monitors the system in “Real-Time”. In 2001, there were 50 attacks available for 9 different operating systems[2]. The uniqueness of this project is the way it generates traffic, it modifies the Linux Kernel that would allow software to emulate many hosts on Internet. The user has complete control to modify and configure the traffic according to the testing systems. This project gives many brilliant ideas that need to be incorporated in to an open source tool.

- **Other Tools:** Authors have mentioned briefly about may other tools, which can be used in any development in this area. NIDSbench, is a NIDS test suite intended for use in testing NIDS and algorithms which assumes that the systems under test are passive. TCPReplay, provide the background traffic by replaying prerecorded traffic. IDSAwake generates false attacks, which resemble well known attacks in order to determine if the systems will produce false alarms. FlameThrower, is load stress tool which identifies network infrastructure weakness, mainly intended to test Firewalls. WebAvalanche, is another stress-testing appliance for web servers. HPING2, is a command-line packet assembler and analyzer wich allows one to create and transmit custom ICMP, UDP, TCP etc. packets.

All these approaches and tools are limited in many ways. And these limitations do not allow them to emerge as independent, open-source, generalized environment or methodology for testing IDS. The authors discuss issues in generating realistic evaluation environments.

4.2 Generating Realistic Environments

There are many issues involved in generating realistic traffic. The first is regarding background traffic which consists of flows without malicious payload. And the second is about the actual traffic which carries attacks for IDS testing. The background traffic contains no harmful data hence it can not be more realistic in testing environment. Though the attack traffic is based on databases which serve as repositories for malicious flows[2]. This database need to be maintained and updated on regular bases. If the environment is to made such that it tests IDS for all attacks, such system will be too expensive. It is recommended to use the security policies already in place as guidelines in the determining which attacks to launch against the IDS.

The paper further suggests on solutions for some of the issues. The issues like Regulatory restriction on user traffic, blocking of firewall, proxy server or ACLs at routers in network for traffic and testing the hosts on which IDS is installed as a victim are some of the major issues in realistic traffic generations. Currently testing is limited to case-by-case scenarios. Some implementations use the real network traffic data as the background traffic though it is not good solution as per the authors. Additionally, it is hard to generate such traffic when network is dynamic.

Recently there are some major approaches which are used to evaluate IDS. DARPA, Custom Software, Audit trails in UNIX and vendor independent lab are some of the important approaches.

- **DARPA Environment:** As discussed in previous sections this environment used signature based system and statistical technique for analysis. This environment was focused on DOS attacks. Advantages of using statistical techniques in IDS over signature based approach and finding out the regions of operations where IDS is not effective were some of the major goals of this project. Test setup consisted components like traffic generation was done same way as in 1999, victim machine was anonymous FTP server and attack injection affected network in scalable and predictable manner degrading victim machine proportional to injecting load.
  This is the implementation of the methodology created in 1999 experiment.

- **Custom Software:** This is the same approach which was followed by the first paper[1] in 1993. In this approach the software platform was developed that simulates intrusions and tests IDS effectiveness[2]. The evaluation criteria is given in the paper as:
  - Broad Detection Range
  - Economy in Resource Usage

  - Resilience to Stress

  The *expect* tool is used to generate traffic along with TCL.

- **Advanced Security Audit Trails Analysis:** This environment is mainly implemented on UNIX systems, where reliability and efficiency of particular one type of Distributed IDS was tested. The positive part of this experiment is, it was tested against various scenarios. Trojan horse, Attempted break-ins, masquerading, suspicious network connection, noising and leakage are some of the attacks it incorporated.

- **Vendor Independent Lab** This last important example is environment created by the NSS group. They perform IDS testing and issue extensive report of it. NSS has established dedicated lab for performing such testing of IDS. This group focuses on the user interface, installation, deployment and management of IDS. This approach of IDS testing is expensive but very accurate since dedicated network is established for IDS where attack traffic generation,system configuration and load generation can be performed in controlled environment. The reporting is also done in very detailed manner. Presently NSS is involved in tests like Attack recognition, performance under load, IDS evasion techniques, stateful performance test and Host performance.

Thus this paper has discussed about the possible ways to build an evaluation environment for IDS testing. It is very prominent in these kind of attempts that none of them have come up with generalized idea of environment which can be applied to any kind of IDS approach. The authors have concluded this paper in many suggestions and requirements for ideal evaluation test-bed for IDS testing.

4.3 Conclusion

The following conclusions have been derived from the extensive study of past attempts to develop evaluation environment and methodology for IDS testing.

- Present approaches to comparative IDS testing and evaluation are inadequate.

- It is not practical to build one single environment which can test IDS against all possible attacks.

- It is best to take live or prerecorded network traffic in realtime from the site where algorithm is been tested.

- DARPA approach was good effort to accomplish this goal though it has many inadequacies like the models for traffic generations were not generalized and the traffic traces gathered after test required lot of rework before it can be used by someone.

- The recent development in attack generation tools are good part of this research. Authors believe that enough sophestication in these tools lead us to al all-one-environment approach in respect to deal with traffic generation problem. Authors have also mentioned such recent try : Thor.

- The new methodology should be able to generate realistic network background traffic as well as ability to merge this traffic with live network traffic.

- Present network simulation tools *e.g. NS* do not incorporate the ability to generate realistic data from standpoint of full traffic generation. Hence more capable open source simulation tools are needed.

- Strictly controlled, distributed in nature, live and realistic traffic generative methodology is needed to achieve the goal of such development.
- Attacks should be maintained in an attack repository such that they may be inserted in to background traffic activity.

The authors emphasize on a great need for an open source traffic generation and attack insertion environment that may be uniformly used for intrusion detection evaluations.

5 Comments on Paper II

As described in detail the second paper provides the final commentary on the status, features and inadequacies of all past and present IDS testing efforts. Also they propose the need and importance of new methodology which is the only possible need for IDS testing. Beyond the scope of this survey there exists one important issue of reducing false positives and misses for any IDS implementation. This issue is also very closely related to the subject of this survey. Until the test reports and results for any newly developed IDS is trustworthy and meaningful in most general respect, one can not determine the accuracy and fidelity of that IDS. Hence better the methodology of testing IDS, better the IDS itself! I would strongly agree on the final comments of the authors of paper 2, which indicates the vital issues regarding building open-source evaluation environment for IDS testing.

References

[1] Puketza, Nicholas J.;Zhang, Kui;Chung, Mandy;Mukherjee, Biswanath and Olsson, Ronald A., *"A Methodology for Testing Intrusion Detection Systems"* IEEE Transactions on Software Engineering, 22, 1996, pp. 719-729.

[2] Athanasiades, Nicholas; Abler,Randal; Levine, John;Owen, Henry; Riley George, *"Intrusion Detection Testing and Benchmarking Methodologies"* First IEEE International Workshop on Information Assurance, 2003.