Attack Recognition for System Survivability: A Low-level Approach

Carol Taylor
University of Idaho
ctaylor@cs.uidaho.edu

Jim Alves-Foss
University of California Davis
jimaf@acm.org

Abstract

This paper extends and builds on previous work that presented a signature-based attack recognition technique. We present general requirements for "survivable attack recognition" and discuss how our approach fits the requirements. Empirical results are given along with an estimate of the measured performance. Other work is reviewed within the context of attack recognition for survivability.

1. Introduction

In 2002, Internet usage in the U.S. continues to grow at the rate of 2 million new Internet users each month [12]. Along with the steady increase in Internet population, comes a corresponding increase in security incidents. Security statistics from Cert [3] indicate the number of incidents reported more than doubled from 21,000 to 52,000 in 2001. These statistics highlight several pertinent facts about today's computing environment relevant to computer security. First, computer security appears to be nearly impossible to achieve given the heterogeneous, decentralized environment of the Internet. Second, insecure computing environments have little effect on user's desire for connectivity. Consequently, securing computers against malicious activity will likely remain an unsolved problem in the near future leaving the majority of users vulnerable to the effects of computer crime.

In recognition of the near impossibility of completely securing computers attached to large, decentralized networks, *survivability* has evolved as a possible solution to this problem. Survivability is defined as the capability of a system to fulfill its mission in a timely manner in the presence of attacks, failures or accidents [5]. Basically, the definition states that all computer systems have an explicit or implicit mission, which must succeed in the face of adverse events. Achieving survivability involves four key properties which include attack resistance, attack recognition, system recovery and system adaptation. Attack resistance defines strategies for repelling attacks such as firewalls and user authentication. Recognition involves identification of attacks and includes techniques like intrusion detection or system logging. Recovery concerns restoration of compromised system data or replacing system binaries. System adaptation is the enhancement of survivability in response to adverse events, which could include system patches or intrusion signature updates [5].

This paper targets attack recognition, as a critical step to system survivability. Intrusion Detection Systems (IDS) also identify attacks but most IDS's simply notify that an event has occurred and do little in the way of active response to intrusions. In contrast to the passive IDS approach, attack recognition in a survivable system is active and can trigger a survivability mechanism in an effort to continue operation. The research presented here is an extension of a previous paper [29] that introduced a technique for attack recognition based on low-level network traffic. As presented, this technique was developed as part of a survivability architecture and is intended to work with complementary tools. Consequently, we focus on a specific set of attacks consisting of those that exploit vulnerabilities in network protocols. We build on the idea of attack recognition for survivability and define requirements for a general solution. We then show how our low-level network approach fits the general requirements for attack recognition for survivability.

The paper is structured into six sections. Section two continues the discussion of survivability and defines a set of criteria for attack recognition for survivability. Sections three and four present an overview of our low-level method from [29] and show how it fits within a survivability framework. Section five examines other solutions to attack recognition. Section six concludes the paper and identifies future research areas.

2. Attack Recognition for Survivability

A system's survivability depends on the system's ability to function in spite of adverse events. While the definition presented previously includes natural destructive events, we focus on human caused events in the form of malicious acts. As presented in the literature, survivability requires the identification of essential and non-essential services [17]. The system is partitioned into essential services defined as those functions which must be maintained in the face of threats (i.e. potential failures) and non-essential services which can be temporarily suspended to allow the system to handle the threat [5]. This approach shifts the focus from maintaining a full set of system services given the presence of attacks, to

Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)
0-7695-1874-5/03 $17.00 © 2002 IEEE

achieving a degraded performance with emphasis on preservation of essential services. For example, given a general purpose workstation, the essential services might be to allow users to complete their work in the form of word processing, spreadsheet use or programming. Non-essential services that could be temporarily suspended include e-mail and Internet connectivity. Thus, the survivability focus is to preserve the user's ability to perform work which would allow temporary disconnection from the network. However, a web server whose primary purpose is delivery of web content would define its essential services as connectivity and content transfer. Consequently, suspension of these services under attack would not be desirable. The definition of the essential services dictates the survivability mechanisms that are employed to mitigate the effects of an attack.

2.1 Attack Recognition Characteristics

In defining characteristics of attack recognition methods for survivability, we need to ask, what are the key requirements for these techniques? A summary of the requirements for attack recognition for survivability is presented in Table 1.

Table 1. Attack recognition for survivability requirements

Recognition Requirement	Benefits
Real Time Performance	Minimize attack damage Reduce recovery cost
Specific Attack Recognition	Survivability mechanism tailored to attack
Low-level Attack Recognition	Pre-empt attack, Minimize attack damage
System Perimeter Recognition	Reduce attack risk Reduce recovery cost

Perhaps the most important requirement is that of real-time or near real-time detection. This requirement is directly related to the desire to minimize attack damage and enhance system recovery. If left unchecked, an attacker can cause more damage which may require total system re-installation. Consequently, IDS techniques based on log file analysis or periodic review of network traffic would not be suitable for attack recognition for survivability.

Another feature important to attack recognition for survivability is specific attack identification. For system survivability, it is important that the type of threat be identified so that the correct survivability mechanism can be employed. This is in contrast to a passive IDS where just knowing an attack is in progress is enough since the typical response is an intrusion alert. Thus, signature based approaches are favored over pure anomaly ID techniques.

The third requirement for attack recognition for survivability is to recognize attacks at the lowest level of system abstraction. We define lower abstraction levels to mean levels that are closer to hardware such as kernel instrumentation [15], system call specification [26], and system call monitoring [10]. Methods based on audit log files are at a higher abstraction level since system events such as user actions and commands are typically what is being monitored. Lower levels allow the possibility of stopping attacks before they complete. One example of this is where a specification approach allows attacks to be intercepted at the system call level [26].

The last key characteristic of attack recognition for survivability is system perimeter identification. This requirement stresses attack identification outside the target system. For example, if the target machine is a workstation on a subnet and the threat is a Denial of Service (DOS)[1] directed at the subnet, the best defense is to stop the DOS far upstream of the victim machine. Thus, there would be little recovery involved of the target machine. For a different attack such as an external buffer overflow, stopping the attack at the perimeter would be preferable to allowing the traffic into the system. A buffer overflow happens so quickly, that it is not always possible to stop this attack once the external traffic enters the system. Ideally, recognizing and filtering the traffic containing the buffer overflow prior to system entry would achieve the greatest benefit for system survivability since there would be no chance of system compromise. Note that it is not possible to stop all attacks outside the system. This requirement is primarily for external attacks that originate outside the system.

3. Signature Based Attack Recognition

In this section we describe a attack recognition for survivability tool that satisfies the four preceding requirements. Most current IDS's use some type of signature to identify malicious activity [1]. Signatures vary widely in both content and complexity depending on the data source. Traditionally, host IDS's have used user profiles or system audit logs to create patterns for later comparison. Securenet [28] and Asax [9] build these types of signatures. Network IDS's contain signatures based on TCP/IP packet headers, packet contents or both. Bro [22], Netprowler [11], and the public domain IDS, Snort [25] use network traffic signatures for ID.

In creating signatures there is a tradeoff between accuracy and efficiency. Including more information in a

[1] A DOS is defined as a denial of service where the attacker's goal is to prevent or deny service for legitimate users by overloading it.

signature increases the accuracy but also increases the amount of processing [14]. This is especially true in network traffic analysis which is required to be fast in order to keep pace with today's fast Ethernet (100Mbps or higher). Highly specific signatures risk generating false negatives since attack variations that affect one or more attributes often bypass the original attack signature [4]. Signature attributes can be extracted from packet headers which leads to fast, efficient processing [20]. Or, signatures can include packet contents which yields more information but at the cost of efficiency. An example of a header attribute would be the number of Syn or Fin flags[2] observed in a one second interval. A packet payload signature might consist of a specific buffer overflow string or more generically an overly long argument to a DNS query, which contains shell code [14].

3.1 Attribute Selection

As previously stated, our goal in the development of an attack recognition method is to identify network attacks in real-time as part of a survivability strategy. In selecting attributes that would distinguish most protocol based[3] attacks from normal network traffic, we tried to identify a minimal attribute set for the highest possible efficiency. Reverse engineering a number of network attacks that target TCP protocol vulnerabilities allowed us to select four attributes from a large potential attribute set. The four attributes consisted of the number of Syn, Fin and Reset flags plus the number of fragmented packets observed per discrete snapshot of network traffic. Surprisingly, these four attributes allowed us to uniquely identify over 20 attacks.

3.2 Model Definition

As defined, we base attack signatures and normal network profiles on frequencies of Syn, Fin and Reset TCP flags and fragmented packets. Let n be the total number of attributes which in our case $n = 4$.

Attack recognition is based on comparisons of attack signatures to a network profile collected in real-time. A *profile* is defined as a vector $P = (f_1, f_2, f_3, ..., fn)$ where f_i, represents the frequency of the i^{th} attribute, $1 \le i \le n$.

We view an attack as a deliberate effort to exploit vulnerabilities in a network protocol. Let t be the total number of attacks in our attack set. Specific attacks will be denoted by A_j where j represents the j^{th} attack and $1 \le j \le t$. We assume that each individual attack, such as a Syn scan, produces a unique set of frequencies that differs noticeably from normal traffic. An attack profile captured in a clean environment, e.g. an isolated network, is saved as a signature. Therefore, a signature of a specific A_j, is defined as the vector, $S_j = (f_1, f_2, f_3, ..., fn)$ where the attribute frequencies are defined as before.

In constructing a normal profile of network traffic, we can either create a profile that is time dependent (i.e. total attribute frequencies per unit time) which creates a variable sized profile or the profile can be defined as a set number of total attribute frequencies. We decided to create profiles of equal size since this simplified both the comparison between attack signature and normal profiles and the real-time monitoring of network traffic. Therefore, we define the size Z of a profile as the summation of the frequencies of the individual attributes, i.e.

$$Z = \sum_{i=1}^{n} f_i$$

It should be noted that Z is not necessarily equal to the number of packets, since a single packet may affect several attributes[4]. For example, in capturing the signature from the *misfrag* attack, if we used a time dependent profile, and we had generated 100 packets with 110 attributes, the total frequency of the profile per second might be 50/sec1 and 60/sec2. For a set profile size, Z, of 100, one full profile would have been created of 100 and a partial profile of 10. Partial profiles are discarded.

3.3 Attack Signature Creation

The goal of attack signature creation is to capture the signature in its purest form. Therefore, the machines used to generate the signatures were set up in isolation of outside network influence. The machines were Linux Pentium PC's set up as attacker and victim. Two popular attack and scanner suites, toast [7] and nmap [8] were used to create attack signatures. Tcpdump [13], a packet capture tool, was run on the victim machine. For creating TCP attack signatures, attacks and scans that target the TCP protocol were selected from toast and nmap.

Attack signatures were created by summing the individual attribute frequencies as described in Section 3.2. Each attack generated a variable amount of traffic ranging from approximately 50 to over 2000 packets. Consequently, choosing the size of a profile, Z, that allows accurate identification of each attack is difficult given the wide range of attack output. If Z is chosen too large, the attack signatures will be obscured by normal network traffic. Yet, if Z is chosen too small, a high false

[2] See our original paper [29] for a discussion of the TCP protocol flags

[3] Results are only reported for the TCP protocol. Both UDP and ICMP protocols are currently under investigation.

[4] Since TCP packets can contain multiple flags, counting the flags produces a flag frequency count which differs from the number of observed packets.

positive rate has been observed, due to the fact that the profile was not large enough to capture the unique attack characteristics.

Experimentation with Z showed that setting Z = 100 produced unique signatures for all of the TCP protocol attacks. However, expansion of the attack set may require a different profile size. Thus, Z remains a tunable parameter.

3.4 Signature Comparison

Complete details of the statistical comparison technique are presented in [29]. An overview of the methodology is included in this section. Under normal operation, a PC connected to the Internet generates traffic from multiple applications. Consequently, we must be able to recognize an attack embedded in a normal stream of network traffic. Ideally, we want the method to be highly efficient so it can function in real-time. The chi-square statistical test [6] was identified as a suitable comparison technique that met our requirements. The chi-square test measures the difference between attribute proportions in two independent samples [6]. Attribute proportions can easily be computed from frequency counts. The chi-square test is computed by the following formula:

$$\chi^2 = 2Z(AD - BC)^2 / (A + B)(C + D)(A + C)(B + D)$$

An attack signature and normal profile can be laid out in a two-by-two contingency table to illustrate the concept. For our case, an example contingency table for the Syn flag attribute of the misfrag attack is presented in Figure 1.

	Syn Flags	Other Flags	
Misfrag Signature	A	B	A+B
Normal Profile	C	D	C+D
	A+C	B+D	2Z

Figure 1. Contingency table for the misfrag Syn flag attribute

A represents the frequency of Syn flags in the misfrag signature and $B = Z - A$. The second row represents the normal profile. C is the frequency of Syn flags in the normal profile and $D = Z - C$. The resulting statistic, χ^2, follows a chi-square distribution and is compared against a threshold value, ε, at a .05 significance level[5]. The threshold value, ε, from the chi-square distribution table at the .05 significance level equals 3.84 [6]. A chi-square value is computed separately for each attribute of an attack signature, A_j. An attack is identified when each attribute from its signature produces a chi-square value below the threshold value, 3.84 in a comparison with the same attribute from a normal profile. If any one attribute has a chi-square value greater than 3.84, the attack is not identified. The expectation is that attacks embedded in normal traffic will still be distinguishable since the attacks create unique attribute patterns.

4. Experimental Results

Preliminary tests of the attack recognition method were conducted by observing normal traffic with specific embedded attacks. Several applications (i.e. Web browser, editor, e-mail program, ftp and telnet) were run on a victim machine, which was then subjected to different attacks. Network traffic was saved to a file for offline processing. Results from four separate attacks including a Syn scan, Fin scan, port flooder (gewse), and fragment attack (syndrop) were reported in [29] (Table 2).

Table. 2 Chi-square values for embedded attacks and normal profiles

Profile	Syn	Fin	Reset	Fragment	Matched Attack
syn scan	.00	.37	.37	.00	nmapsS
fin scan	.00	.00	.02	.00	nmapsF
syndrop	.00	.00	.00	1.63	syndrop
gewse	.00	.00	.00	.00	gewse
normal	15.00	.00	21.30	.00	none

All of the attacks were correctly identified as matching their respective signatures. In tests of hundreds of normal profiles, no false positives were observed. A zero indicates that the attribute proportion was identical to that observed in the attack signature. An example normal profile is included in Table 1, which illustrates that two out of the four attributes did not match any of the attacks.

Measurements were run to compute the signature analysis overhead. The machine consisted of a Pentium PC running at 1 GHz. Tests of 6, 12, 24, 43 and 102 attack signatures were conducted for comparing a normal profile of size, $Z = 100$. With 6 signatures in the attack set, the total comparison time was 300 microseconds. For 12 signatures, the processing time was approximately 400 microseconds. The time increase appeared to be linearly scaled with the number of signatures and was 600, 900

[5] A .05 significance level is a typical level for statistical tests found in scientific literature and generally represents a minimum level required for statistical significance

and 1,900 microseconds respectively for 24, 43 and 102 signatures. Signature processing overhead on the machine was measured at < 1% of the total running system. Consequently, the method should scale well to a larger signature set.

4.1 Recognition of Attacks

This approach appears to have promise as a light-weight network ID technique. Attack recognition was accurate in initial system tests with no false positives. However, since our interest is in recognition techniques for system survivability, we ask how well does it satisfy the four requirements for attack recognition for survivability?

The first stated requirement was real-time attack recognition in order to minimize system damage. While the method was not tested in real time, calculation of the estimated off-line operating time showed that the approach could easily keep pace with current network traffic speeds.

The second requirement for attack recognition for survivability was accurate attack identification. Our technique, being signature based, identifies specific attacks, which would permit survivability responses based on the attack identified.

The third requirement was low-level abstraction which was satisfied by our tool since it monitors individual network packets. Monitoring network packets is considered a low-level approach in the IDS community [20].

The last requirement for perimeter attack recognition is satisfied since we expect a network attack recognition component to be located outside of a target machine. Ideally, we would place the attack recognition component outside the system either before or after a firewall or router as another defensive layer. Firewalls and routers can be adaptively configured in response to an attack. Thus, damaging packets could be filtered prior to their reaching the target system. Other survivability mechanisms could be triggered depending on the attack. For instance, previous work demonstrated an agent based survivability mechanism that stopped a smurf attack by resetting the router table in real-time [16].

5. Related Work

Intrusion detection has been an active research area for nearly two decades producing many innovative approaches to the detection of malicious activity. Among the general purpose solutions, a few are particularly well-suited to attack recognition for survivability and meet the criteria outlined in Section 2. These systems are reviewed and contrasted with our approach.

Snort is a public domain IDS which offers real-time network attack recognition [25]. Snort meets most of our requirements for attack recognition for survivability since it operates in real-time, is signature-based for exact attack identification, performs low-level traffic analysis and offers perimeter defense for network based attacks. Snort performs packet payload inspection, which tends to slow it down but also allows identification of more attacks. However, Snort is a standalone ID system and does not attempt to coordinate with other defense mechanisms such as host based detection or intrusion response. Our system was designed as one component in a coordinated survivability architecture with an emphasis on highly efficient processing. We expect this component to work with other defensive mechanisms that cover attacks outside the scope of this tool.

Bro is another recent IDS and was developed to monitor a high-speed gateway for 1000's of machines. Bro includes a specialized language for security policy definition and uses network traffic based signatures for intrusion detection [22]. This system also meets most of the requirements for attack recognition for survivability. Bro operates in real-time, uses low-level traffic signatures and serves as a perimeter defense mechanism. Similar to Snort, Bro is a complete IDS which doesn't try to coordinate with other defensive layers.

In [23], a combination of signature and anomaly detection is applied to network traffic. The signature detection allows specific identification of attacks while anomaly detection provides general notification for traffic conditions that fall outside of a historical norm. This approach operates in real-time and provides a perimeter type of defense. While the data source is low-level, signatures consist of expert system rules which tends to slow down detection. This method operates as a component of the distributed Emerald IDS [19] which makes this approach the most conceptually similar to ours.

A host-based technique for performing low-level attack recognition was presented in [15]. This technique uses kernel module frequencies to construct attack signatures. The method performs real-time attack recognition, and operates at an extremely low-level of abstraction. A perimeter type of defense is not possible but the technique was designed as part of a survivability architecture and it is expected to work in concert with other defensive tools.

Other research combines attack recognition and attack resistance. These methods are worth mentioning since they are specifically aimed at enhancing system survivability. In [2], intrusion detection is integrated in with a web server application. Consequently, the IDS can pre-empt malicious acts since intervention can be placed at any point in the HTTP request processing cycle. Though limited to a specific application, overall system

survivability is increased by stopping attacks that exploit web server vulnerabilities but threaten the entire system. Two other methods use system call data to prevent completion of malicious system calls. Forrest extended the ID work in [10] to monitor processes that appear to behave maliciously [27]. The idea is to abort or delay system calls in processes that appear malicious so that system damage is prevented. Constant monitoring of the processes in real-time appears to be feasible producing manageable overhead. A similar approach is presented in Sekar et al [26] who developed a technique to intercept and validate system calls and their parameters. Their approach is described as a specification based method since they specify correct behavior for system calls. Calls that appear to violate the specification are not allowed to execute.

6. Conclusion and Future Work

This paper extended our previous work with a low-level network attack recognition method. We defined general requirements for attack recognition that functions as a survivability mechanism and show how our method satisfies these requirements. The work presented is preliminary and describes the technique showing results from a limited number of attack signatures.

Further work will expand the creation of attack signatures by extending the protocols recognized to include UDP and ICMP. This work is currently being implemented. Other attributes than TCP flags and fragmented packets will need to be identified for creating the attack signatures. Since testing to identify false positives was not extensive, additional tests with a much large application set will be necessary before we can determine realistic false positive or false negative rates. Coordinating this component with other survivability tools is another future task. Network attack recognition allows a specific set of attacks to be identified but it is not a comprehensive solution. Pairing network and host attack recognition should allow the tools to complement each other. In a truly survivable system attack recognition is considered to be one component along with resistance and recovery. Recognition of attacks should generate a system response. However, it is not always clear exactly what the response should be. Response types that enhance survivability given specific classes of attacks will be another area of future research.

References

[1] Allen, J. et al, "State of the practice Intrusion Detection Technologies", Carnegie Mellon, SEI, Tech Report, CMU/SEI-99-TR-028, ESC-99-028, January 2000.

[2] Almgren, M. and U Lindqvist, "Application-Integrated data collection for security Monitoring", In Proceedings of the 4th International Workshop on Recent Advances *in Intrusion Detection,* Oct, Davis, California, 2001.

[3] Cert, www.cert.org, 2001.

[4] M. Cooper, S. Northcutt, M. Fearnow, K. Frederick, ***Intrusion Signatures and Analysis***. New Riders, Indianapolis, IN, 2001.

[5] Ellison, R. J., D. A. Fisher, R. C. Linger, H. F. Lipson, T. A. Longstaff, N. R. Mead, "An Approach to survivable systems", *NATO 1st Symposium on Protecting Inform. Systems in the 21st Century,* Wash. D.C., Oct. 25-27, 1999.

[6] Ferguson, G. A. ***Statistical Analysis in Psychology and Educatio,.*** McGraw-Hill, New York, N.Y. 1981.

[7] Fyodor, ***nmap*** , www.insecure.org/nmap, 1999.

[8] Gridmark,***toast,*** Packetstorm.securify.com/DoS/indexsize.shmtl, March, 2000.

[9] Habra, N., B. Le Charlier, A. Mounji, I. Mathieu, "Asax: Software architecture and rule-based language for universal audit trail analysis in: Y. Deswarte, G. Eizenberg, J.J. Quisquater (Eds*.) Proc. 2nd European Symp. on Res. in Comp. Sec. (ESORICS), Toulouse, Berlin, Lecture Notes in CS, Vol. 648,* Springer Verlag, Nov. 1992.

[10] Hofmeyr, S. A., and S. Forrest. "Intrusion Detection Using Sequences of System Calls", *Journal of Computer Security*, Vol. 6, pp. 151-180, 1998.

[11] Hurwitz Group, "Hurwitz Report: Axent Technologies' NetProwler and Intruder Alert", Hurwitz Group, Framingham, MA, Sept. 2000.

[12] Internet News, "U.S. Internet population continues to grow", www.internetnews.com/IAR/print.php/96954/, Feb.6 2002.

[13] Jacobson, V., C. Leres, and S. McCanne, ***tcpdump,*** LBNL, University of California, June 1997, ftp://ftp.ee.lbl.gov/tcpdump.tar.Z.

[14] Kent Frederick, K. "Network intrusion detection signatures", http://online.security.focus.com/infocus/1524, 2002.

[15] Krings, A.W., S. Harrison, N. Hannebutte, C. Taylor and M. McQueen, "Attack Recognition Based on Kernel Attack Signatures, In *Proceedings International Symp. on Information Systems and Eng., (ISE'2001),* Las Vegas, June 25-28, 2001.

0-7695-1874-5/03 $17.00 © 2002 **IEEE**

[16] Krings, A.W., S. Harrison, N. Hannebutte, C. Taylor and M. McQueen, "A two-layer approach to survivability of networked computer systems", *Proceedings 2001 Int. Conference on Advances in Infrastructure for Electronic Business Science and Education on the Internet (SSGRR 2001),* L'Aquila, Italy, August,2001.

[17] Linger, R. C., N. R. Mead, H. F. Lipson, "Requirements Definition for Survivable Network Systems", www.cert.org/research/papers.html.

[18] Lindqvist, U. and P. A. Porras, "Detecting Computer and Network Misuse Through the Production-Based Expert System Toolset (P-BEST)", In *Proceedings of the 1999 IEEE Symposium on Security and Privacy, Oakland, Calif.,* May 9-12, 1999.

[19] Neumann, P. G. and P. A. Porras, "Experiences with Emerald to Date", *Proceedings of 1st Usenix Workshop on Intrusion Detection and Network Monitoring,* Santa Clara, CA, Apr. 11-12, 1999

[20] Northcutt, S., V. Irwin, B. Ralph, ***Shadow***, Naval Surface Warfare Center, Dahlgren Lab, 1998.

[21] Northcutt, S., ***Network Intrusion Detection: An Analysts Handbook,*** New Riders, Indianapolis, IN, 1999.

[22] Paxon, V., "Bro: A system for detecting network intruders in real-time",*Computer Networks, 31(23-24):2435-2463*, 1999.

[23] Porras, P. A. and A. Valdes. "Live traffic analysis of TCP/IP gateways", In *Proceedings of the 1998 ISOC Symp. on Network and Distirbuted Sys. Sec.*, San Diego, CA, March 11-13, 1998.

[24] Ranum, M. J., K. Landfield, M. Stolarchuk, M. Sienkiewicz, A. Lambeth, E. Wall, "Implementing a Generalized Tool for Network Monitoring", *Proceedings of 11th Syst. Admin. Conf. (LISA 97)*, San Diego, CA, Oct. 1997

[25] Roesch, M., "Snort – Lightweight Intrusions Detection for Networks", www.clark.net/~roesch/security.html

[26] Sekar, R., Y. Cai, M. Segal, "A Specification-Based Approach for Building Survivable Systems", In *Proceedings 21st National Information System Conference,* Arlington, VA, Oct. 6-9, 1998.

[27] Somayaji, A., and S. Forrest, "Automated Response Using System-Call Delays", In *Proceedings of the 9th Usenix Security Symposium, August 14-17*, 2000.

[28] Spyrou, T., J. Darzentas, "Intention Modeling: Approximating Computer User Intentions for Detection and Prediction of Intrusions, in: S.K. Katsikas, D. Gritzalis (Eds.), *Information System Security*, Samos, Greeze, May 1996, pp. 319-335.

[29] Taylor, C. Krings, A.W., S. Harrison, N. Hannebutte, M. McQueen, "Low-Level network attack recognition: a signature-based approach", In Proceedings of 13thIASTED Int. Conference on Parallel and Distributed Computing and Systems (PDCS 2001), August, 2001, Anaheim, California.