Shifting the Focus of Survivability: Back to the Basics*

A.W. Krings, W.S. Harrison
Computer Science Dept.
University of Idaho, Moscow ID, USA
{krings,harrison}@cs.uidaho.edu

M. McQueen, S. Matthews
INEEL
Idaho Falls, ID, USA
{amm,sxm}@inel.gov

Abstract

This research introduces a new paradigm to survivability. The philosophy of the approach is to consider a hierarchical solution space, where survivability features for specific attacks are applied at the lowest suitable level. Whereas the hierarchy as a whole is assumed to represent a comprehensive solution, each level is limited in scope to deal with attacks which have specific characteristics. Some attacks have such evident characteristics at a specific level, that survivability features can be applied with very high degree of certainty.

The focus of this paper is on solutions at the lowest level of complexity. Specifically, we address the strength of signature based attack recognition and recovery at the kernel level. This method has been successfully demonstrated in a survivability architecture based on the Linux operating system.

1 Introduction

The research presented here attempts to break away from comprehensive, monolithic survivability mechanisms and subscribes to a highly focused hierarchical approach to survivability of networked computer systems.

We adopt the definition of survivability given in [3], which was formulated with respect to *Resistance*, *Recognition*, *Recovery*, and *Adaptation*, with research emphasis on Recognition and Recovery, as discussed below.

Recognition: It is crucial to recognize whether a computer has been compromised or is under attack. This has been the focus of much research in intrusion detection (ID) [1, 12] and has resulted in a variety of intrusion detection systems (IDS), including projects like EMERALD [10] or NetSTAT [9]. However, the objective of an IDS is usually recognition and not survivability, e.g. reactionary measures or recovery.

Recovery: A key characteristic of survivable systems is the ability to mask or restore essential services, often referred to as *critical functionalities* [11]. Approaches to survivability of critical functionalities have been considered at different levels. In [4] a top-town survivable network analysis method based on the concept of a survivability map was presented. This map considered intrusion scenarios, their resistance, recognition and recovery strategies at a high level. Low level, e.g. kernel signature based, approaches have been introduced by [2, 7, 8].

First, we wish to put this research into the proper context. Considering a class of attacks, the question may arise "What is the lowest level of complexity at which these attacks exhibit distinct, observable characteristics"? The two key issues are the level of complexity and the characteristics of the attack class. With respect to the level, it can be expected that lower complexity solutions translate into higher real-time potential. Considering attack characteristics, one may assume that at low levels certain attacks, e.g. DDoS, are very recognizable, whereas other attacks, e.g. stealth attacks, fail to produce distinctive characteristics. This suggests a hierarchy of levels, each one considering attacks that will produce distinctive characteristics at that specific level. It is advantageous to deal with each attack at the lowest possible level. This is similar in nature to a filtering process based on complexity.

While most research efforts tend to concentrate on high level solutions, e.g. intrusion detection based on audit trails, we focus on the effectiveness of solutions situated at the lowest possible level and incorporate the basic survivability model described in [8].

*This work has been supported by the Idaho National Engineering Laboratories (INEEL).

2 Motivation, Background and Assumptions

A core assumption of this research is the notion of a *standard user environment*. We view such an environment as a collection of typical powerful desktop computers, operated mostly by single individuals. The usage of these "dedicated" workstations is in general very low. Most common user profiles include applications such as x-windows, browsers, email, compilers, or small web servers. However, unlike systems such as transaction systems or main servers, the actual utilizations are generally surprisingly low, as can be verified on individual systems using utilities such as `top`.

2.1 Passive and Active Survivability

Similar to the concepts of passive and active redundancy in fault-tolerance, we partition survivability based on the characteristics of the solution space. Specifically, we assume that *Passive Survivability* employs implicit techniques, such as authentication or fault masking using redundancy. *Active Survivability*, on the other hand, requires recognition of events, e.g. malicious attacks, as the first step in a sequence of steps leading to recovery.

This research focuses on Active Survivability. At the lowest level of consideration is attack recognition. Recognition is based on the detection of suspicious or abnormal behavior. Two basic strategies are used to analyze both known and unknown attacks. The first is based on attack signature detection, the second considers anomaly detection. Ideally, attack recognition based on attack signatures requires a priori knowledge of *all* attack scenarios, whereas anomaly detection requires full knowledge of the expected behavior of the system to detect all attacks. Neither of these strategies can achieve complete realistic detection coverage [1]. Our emphasis is on attack recognition based on attack signatures.

2.2 Levels of Abstraction

Attack recognition is based on signatures reflecting the characteristics of the attack. Signatures can be defined at different levels of complexity. Most research and projects involving signatures has been based on discrete events that are derived from system log files. These data are, however, at a relatively high level of abstraction, since log files record only specific audit events. At a lower level of complexity one can derive signatures of system calls. Such an approach has been taken in [5, 14].

This research considers kernel level signatures. Kernel based attack detection is positioned at the lowest level of observation, just above hardware assisted detection. The benefits of kernel based intrusion detection or attack recognition have only been discussed in the literature in the recent past. Some kernel based intrusion detection systems are based on wrappers [6], or on dynamic profiling [2]. Our research is based on signatures derived from kernel level functional profiling.

Figure 1 displays the relationship of signatures and their analysis at different levels of abstraction. First, kernel based signatures are at the lowest level that can be controlled by software. This level is considered in [2] and [7]. Next, system call logs allow profiling at a higher level. Thus signature generation is at a much coarser granularity, as considered in [5] and [14]. Finally, at the highest level are signatures derived from log file manipulation. Most IDS research is based on log file analysis [1]. The level of abstraction of the signatures is inversely proportional to the potential real-time feasibility of recognition and reaction to attacks.

The different levels of abstraction can be mapped into different levels of a survivability hierarchy. Solutions at the lowest level not only promise high real-time feasibility, but also allow filtering out attacks, rather than shifting the responsibility to higher levels. Thus, higher levels can focus on attacks requiring more complex analysis.

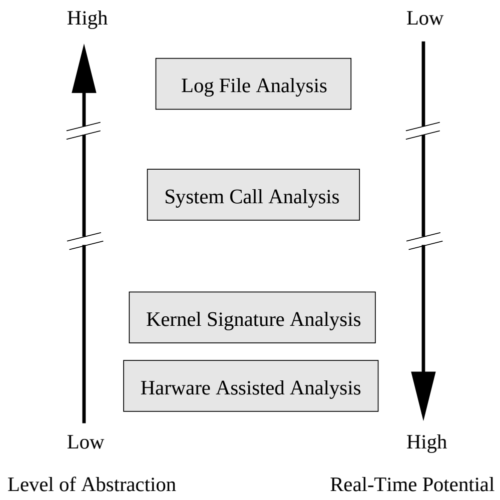

Figure 1: Levels of Abstraction and Real-Time Potential

2.3 Clean Measurements

System survivability can be seen from two different viewpoints. One can look at the architecture as the result of an off-line design process, or one can focus on on-line, real-time protective capabilities. The off-line approach focuses on designing an architecture based on general information of attack scenarios and protection of critical functionalities specified a priori. On-line survivability takes advantage of information gathered at run-time, which allows the architecture to adapt itself to new attack scenarios or changes in the overall computer and network environment.

With respect to the attack signature generation process, we are attempting to turn the attack recognition problem into a basic dynamic measuring problem. At the heart of the approach is the process of clean measurement. We want to measure the impact of specific attack scenarios on the system as accurately as possible off-line. Once the measurements are taken and "cleaned up" they can constitute a frame of reference in on-line attack recognition.

In [7, 8] it was demonstrated that one can achieve attack recognition, independent of application environments, based on the quality and analysis of the attack signatures alone. The measurement procedure is modeled on measurements of physical phenomena. First, the focus of attention was on the network portion of the target system, a Linux based host. Next, in order to reduce system noise, all measurements were conducted off-line, in an isolated network, eliminating any unrelated network activity. There were no applications executing, i.e. the systems were running as console (without x-windows support). Lastly, signatures were extracted from an absolutely idle system as it was subjected to very narrowly defined attacks.

Whereas the signature generation process above is invoked off-line and in a clean environment, attack recognition is an on-line process, i.e. it is done in real-time. An off-line analysis determines the usefulness of the signature in identifying critical functions, and investigates their relationship to other, similar, attack signatures. It is only during run-time, in the on-line process, that we attempt to recognize the attack signatures in the noisy overall profile of the networked system. It is very important that the reader be clear about the differences of these two cases.

3 Survivability Architecture

Similarly to [2], we view the system as a collection of functionalities. The functionalities are observed during specified time intervals Δt. Specifically, we view a system in terms of its system profile $P_{sys}(\Delta t)$, which is composed of the profiles of all functionalities $P_i(\Delta t)$ executing during Δt. Thus, for any Δt we have

$$P_{sys}(\Delta t) = \sum_{i=1}^{k} P_i(\Delta t)$$

where k is the number of functionalities active during the time interval. Each $P_i(\Delta t)$ is a vector of a length equal to the number of identities F, i.e. C functions, profiled. Therefore, if there are n identities profiled, then

$$P_i(\Delta t) = (f_1(\Delta t), f_2(\Delta t), ..., f_n(\Delta t)),$$

where $f_j(\Delta t)$, $1 \leq j \leq n$, is the number of times function F_j has been called (or activated) during Δt.

We assume attacks to be atomic. A specific attack i, denoted by A_i, is the smallest attack technology unit, e.g. a port sweep. Limiting the scope of an attack to atomic units allows us to focus on a very narrow sets of affected functions in the operating system (OS), application and network.

Profiles during atomic attacks are of special interest and result in attack signatures. An attack signature S_i is the portion of a profile that is attributable to an attack A_i. Only non-zero profile components are considered. Thus,

$$S_i = (f_{\alpha(1)}(\Delta t), f_{\alpha(2)}(\Delta t), ..., f_{\alpha(s_i)}(\Delta t)),$$

where α is a function that maps (one-to-one) the indices of S_i to the indices of the functions profiled. Note that s_i, the length of vector S_i, is signature dependent.

At runtime, signatures are used to identify specific attacks. The analysis consists of simple vector comparisons [8, Lemma 2]. Individual signatures are collected in a signature library. At runtime, profile $P_{sys}(\Delta t)$ is compared with all signatures S_i of the library.

3.1 Reactionary Mechanisms

Recovery or reactionary mechanisms can be located at different levels. At the lowest level, survivability handlers can be inserted into the kernel itself. As such, these handlers operate at the lowest level of complexity and also have the fastest possible response time. On the other hand, signature analysis may be used to trigger autonomous response agents, capable of employing high level response mechanisms. Such a system has been successfully demonstrated to surviving distributed denial of service attacks in [8].

4 Extending the Model

In the above discussion, only the network portion of the OS was considered and signatures S_i were defined. However, attacks may not produce identifiable network signatures, but stimulate activities elsewhere in the OS. The concept of signatures can be extended from simple S_i to encompass N-version dissimilarity. With respect to the network portion of the OS considered above, S_i will be denoted by S_i^{NET}. In general, the association of the signature is indicated by the superscripts. Signatures based on IP protocol flags, S_i^{IP}, were introduced in [13]. When combined, S_i^{NET} and S_i^{IP} allow a two-version dissimilar approach to attack recognition.

An alternative approach to widen the scope of signatures is signature partitioning. Specifically, we are currently investigating partitioning based on functionalities, e.g. network, file system, memory management, etc. Initial experiments with signatures spanning the entire kernel, rather than a specific portion, indicated that the profile attributable to the attack is likely to be hidden within the overall system activities. Partitioning, in principle, aids in the detection of attacks targeting specific portions of the OS, within the respective partition. For example, a buffer overflow attack may not result in a distinguishable networks signature S_i^{NET}, but may generate a well defined memory management signature S_i^{MM}.

Using N-version dissimilarity and signature partitioning, we hope to increase the effectiveness of surviving a larger number of attacks.

5 Conclusion

We propose to address survivability with respect to different levels of complexity. The philosophy is to recognize specific attacks at the lowest suitable level. The levels are narrowly defined in focus. At the lowest level, our research with simple signatures based on the network portion of the operating system has shown that attack recognition and recovery can be achieved effectively in real-time. This theme can be extended by the principle of N-version dissimilarity and signature partitioning in order to consider different classes of attacks.

References

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

[2] S. Elbaum and J. Munson, *Intrusion Detection Through Dynamic Software Measurement*, Proc. Eighth USENIX Security Symposium, 1999.

[3] E. Ellison, L. Linger, and M. Longstaff, *Survivable Network Systems: An Emerging Discipline*, Carnegie Mellon, SEI, Technical Report CMU/SEI-97-TR-013, 1997.

[4] E. Linger and M. Longstaff, *A Case Study in Survivable Network System Analysis*, Carnegie Mellon, SEI, Technical Report CMU/SEI-98-TR-014, 1998.

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

[6] K. Calvin, et. al., *Detecting and Countering System Intrusions Using Software Wrappers*, Proc. 9^{th} USENIX Security Symposium, 2000.

[7] A. Krings, W. Harrison, et. al., *Attack Recognition Based on Kernel Attack Signatures*, Proc. International Symposium on Information Systems and Engineering, Las Vegas, pp. 413-419, 2001.

[8] A. Krings, W. Harrison, et. al., *A Two-Layer Approach to Survivability of Networked Computing Systems*, Proc. International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, L'Aquila, Italy, pp. 1-12, 2001.

[9] G. Vigna and R. Kemmerer, *NetSTAT: A Network-Based Intrusion Detection Approach*, Proceedings of the 14 th Annual Computer Security Applications Conference, Scottsdale, Arizona, December 1998.

[10] P. Neumann and P. Porras, *Experience with EMERALD to DATE*, Proc. 1st USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, California, pp. 73-80, 1999.

[11] P. Neumann, *Practical Architectures for Survivable Systems and Networks*, (Phase-Two Final Report), Computer Science Laboratory, SRI International, June 2000.

[12] Purdue University, *Coast intrusion detection*, http: //www.cerias.purdue.edu/coast/intrusion-detection/ids.html.

[13] C. Taylor, et. al., *Low-Level Network Attack Recognition: A Signature-Based Approach*, Proc. 13^{th} International Conference on Parallel and Distributed Computing and Systems, Anaheim, California, pp. 570-574, 2001.

[14] C. Warrender, et. al., *Detecting Intrusions Using System Calls: Alternative Data Models*, IEEE Symposium on Security and Privacy, pp. 133-145, 1999.