The Inquisitive Sensor: A Tactical Tool for System Survivability*

Ulf Lindqvist
System Design Laboratory
SRI International
333 Ravenswood Ave
Menlo Park CA 94025
ulf@sdl.sri.com

Abstract

This paper proposes methods by which traditionally passive intrusion detection sensors can be instrumented to take certain tactical actions in order to increase the certainty by which they can make decisions. Examples show how a sensor can request injection of additional data into the event data stream it is monitoring, as a reaction to its observation of incomplete indications of threats. As the threat level increases, the sensor could increase its degree of activeness and thereby its visibility. When a system is under attack, an accurate and timely situation assessment enabling effective countermeasures is probably more important to system survivability than keeping the sensors concealed.

1. Background

In the EMERALD project, we have developed a number of intrusion detection (ID) sensors for different analysis targets, such as network protocol data from different communication layers, host audit data, and so forth. Also, we use different analysis techniques, such as rule-based forward-reasoning expert systems [1] and probabilistic methods [4]. We strongly believe in the advantages of this two-dimensional diversity (target and technique) for ID sensors, especially with respect to the resulting combined coverage. In this paper, an ID *sensor* is considered to consist of both an event data collection component (CIDF: E-box) and an analysis component (CIDF: A-box [3]).

*This work is part of a project currently funded by the Advanced Technology Office of the Defense Advanced Research Projects Agency (DARPA/ATO), under contract number N66001-00-C-8058. Distribution Statement 'A', Approved for public release—Distribution Unlimited. The views presented herein are those of the author and do not necessarily reflect the views of the supporting agency.

One common theme for all these sensors is that they passively monitor a data stream, and the only external action they are capable of taking is the transmission of an alert message. That alert message is intended to be picked up by another component that could, for example, present it to a human operator, store it in a database, correlate sensor messages in higher-level analyses, and/or issue automatic response actions. One advantage of this passive monitoring is that the sensors function equally well in batch (offline) mode and in real-time (online) mode. Although most sensor deployment concerns real-time operation, batch mode is useful for testing, evaluation, and after-the-fact (forensics) analysis. Another advantage is that for certain event streams, such as network data, a passive sensor could be made completely invisible and undetectable (provided that an attacker cannot observe the sensor output channel).

However, if we are ready to give up the requirement that sensors produce the same results in batch mode as in real-time mode, we could let the sensors take advantage of the fact that they are operating in a real-time situation. This paper presents ideas on how sensors could be instrumented to perform simple actions that could significantly improve the correctness of their situation assessments. In a situation where we suspect that a system is under attack, allowing sensors to become visible in order to make better response decisions could be a reasonable tactical trade-off.

2. Queries to confirm compromise

We present examples of how the sensor could use a simple query to dramatically improve the confidence level of an alert or, in cases where the result of the query does not confirm the suspicion, cancel the issuing of an alert that would have been a false positive.

In all cases presented, we can cause the data resulting from the sensor's query to be injected into the event stream

In Supplement of the 2001 International Conference on Dependable Systems and Networks, pages C-14–C-16, Göteborg, Sweden, July 1–4, 2001.

it is monitoring. Thus, the sensor will still receive all its input data in the original input channel. This will also enable batch mode analysis of data collected while an active sensor was present, as the additional data will be present in the batch file. However, we must be aware of the risk that the attacker also can observe (and, in worst case, manipulate) the query and/or the result and thereby infer information about the sensor's existence and its knowledge. In particular, this risk is high when the event stream represents network traffic.

2.1. Host: Network interface status

This example is based on our experience with the Solaris BSM event stream. Among many other things, we want a host-based sensor to raise an alert when the network interface card is set in promiscuous mode, as this indicates that a network sniffing process was started on the host—a typical action for an attacker who has taken control of a host in a network and wants to collect information for further penetrations.

The problem is that the promiscuous mode is set with arguments to a *putmsg()* system call, and those arguments are not provided in the event stream. Even if they were, analysis of all *putmsg()* calls would probably be too resource-intensive for a lightweight host-based sensor.

Instead, our sensor looks for an *open()* call to the network device, a call that must precede the *putmsg()* described above. However, there are many other reasons for opening the network device than to put it in promiscuous mode, and therefore an alert based solely on the *open()* call risks being a false positive.

Suggestion: It is relatively easy to query the network interface to determine whether or not it is in promiscuous mode. Such a query could be triggered by the sensor after it has observed the *open()* call described above, leading to an alert only if the interface really is in promiscuous mode.

2.2. Host: Critical local processes

Some attacks aim at killing security-critical processes, such as logging and reporting daemons, authentication services, and so forth. As many or all of these critical processes are started before a host-based ID sensor is started, it is very difficult for a passive sensor monitoring an audit trail to determine whether those processes are running at any given time. The fact that in an audit trail such as the Solaris BSM data, processes are identified only by process ID and not by command name, makes it even more difficult to identify those critical processes. Even if the sensor sees information that a process with a certain process ID exited or dumped core, at that point it is too late to identify the program image that corresponded to that process.

Suggestion: When starting, the sensor could issue a query about what processes are running on the system and compare that information to an operator-supplied list of critical processes. It could repeat this query periodically and/or when it suspects that a critical process was killed, and then alert when one of those processes is no longer running. It could also use this information to connect the process ID in an *exit()* record (or other event records) to a program name.

2.3. Network: Service availability

We have a network-based sensor using probabilistic analysis technology [4] capable of service availability monitoring (also known as "blue sensor"). It can determine, with a certain *a posteriori* probability, that a monitored network service (such as a Web server) appears to be down because client requests fail with an unusually high frequency. The cause could be a denial-of-service attack or other security problem, or an inadvertent event such as a hardware failure, misconfiguration, or a software bug in the server operating system or application.

Suggestion: When the sensor suspects that a service may be unavailable, it could trigger its own request for service and measure the response time. If there is no response or the response time is longer than some limit, the sensor now has more confidence in a "service down" alert. On the other hand, a healthy response indicates that the service itself is working, but there could be another reason why client requests fail. Either way, the additional information helps in diagnosing the situation. This is an example of the integrity checking probe suggested in [2].

In this case, the sensor can stay invisible if it has an out-of-band communications channel (with respect to the event stream it analyzes) to a simple component that it can order to issue the probing request. As the probe and the reply from the service (if one occurs) will be in the event stream, observable to the sensor and possibly also to the attacker, we need to masquerade the request so that it looks innocuous.

2.4. Network: Configuration discovery

In our current sensor design, it is left to the operator to either judge the relevance of an alert, or to manually configure the sensor so that it can make relevance judgments. Knowledge that affects alert relevance could, for example, concern target vulnerability, which depends on network topology, target configuration, and target platform (attempts to exploit a Windows vulnerability on a Unix host might be irrelevant and vice versa). Manual entering of this information is error prone and can easily become incomplete, dated, or otherwise incorrect.

Suggestion: Instead of making these judgments manually outside or inside the sensor, the sensor could trigger a configuration scan (such as the *nmap queso* scan) of the monitored targets, automatically collect the required knowledge, and use it to determine alert relevance when attacks are detected. This could reduce the risk of misconfiguration or omissions to update the configuration. A problem with this approach is that other sensors could observe the scan and raise alerts about it. In addition, if it is done as an immediate reaction to an observed attack, it could disclose the existence of the sensor to the attacker.

3. Dynamic level of activeness

An efficient way to make life difficult for an intruder is to make your environment unknown and unpredictable. If the attacker does not know what sensors and countermeasures you have in place, it is harder for him to circumvent them. Therefore, keeping sensors concealed and "stealthy" is often a good initial tactic, and any active behavior should be carefully crafted not to reveal any information to the attacker.

However, when the system is under attack and initial countermeasures do not seem to help, we suggest increasing the degree of active sensor behavior along with other escalated responses. At this point, the sensor input data could be less reliable because the attacker might be in control of parts of the environment, and active queries could help in confirming or denying indications from the regular data source. Also, as the issued countermeasures have already given away some information about the presence of sensors, the benefits of enabling more accurate decisions in an already dangerous situation could outweigh the risks.

4. Discussion

From a system survivability perspective, it is important to have tools on a tactical level that can autonomously and dynamically adapt their behavior to threats to the system. We have presented the concept of an inquisitive sensor that can dynamically augment the event data stream it is monitoring, by causing queries to be issued and the replies to be injected into the event stream.

In this paper, we have focused on the individual sensors, and have not discussed sensor coordination and strategic issues. It could be argued that the limited view of a low-level sensor is insufficient to make decisions about any active behavior, but that depends on the monitored system and the current threat level.

We plan to augment some of our currently passive sensors with active capabilities and conduct experiments to test and refine the ideas presented here.

Acknowledgments

Thanks to Ken Theriault and other members of the DARPA IA and AIA communities for comments on an earlier version of this paper.

References

[1] U. Lindqvist 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*, pages 146–161, Oakland, California, May 9–12, 1999.

[2] P. A. Porras and A. Valdes. Live traffic analysis of TCP/IP gateways. In *Proceedings of the 1998 ISOC Symposium on Network and Distributed Systems Security*, pages 142–154, San Diego, California, Mar. 11–13, 1998.

[3] B. Tung. The Common Intrusion Detection Framework (CIDF), June 22, 1999. *http://gost.isi.edu/cidf/*.

[4] A. Valdes and K. Skinner. Adaptive, model-based monitoring for cyber attack detection. In H. Debar, L. Mé, and S. F. Wu, editors, *Recent Advances in Intrusion Detection (RAID 2000)*, volume 1907 of *LNCS*, pages 80–92, Toulouse, France, Oct. 2–4, 2000. Springer-Verlag.