Detection of Intrusion Across Multiple Sensors

William Long[**a], Jon Doyle[b], Glenn Burke[a], and Peter Szolovits[a]
[a]Laboratory for Computer Science, Mass. Inst. Of Technology; [b]Dept. of Computer Science, North Carolina State University

ABSTRACT

We have been developing an architecture for reasoning with multiple sensors distributed on a computer network, linking them with analysis modules, and reasoning with the results to combine evidence of possible intrusion for display to the user. The architecture, called MAITA, consists of monitors distributed across machines and linked together under control of the user and supported by a "monitor of monitors" that manages the interaction among the monitors. This architecture enables the system to reason about evidence from multiple sensors. For example, a monitor can track FTP logs to detect password scans followed by successful uploads of data from foreign sites. At the same time it can monitor disk use and detect significant trends. A monitor can then combine the evidence in the sequence in which it occurs and present evidence to the user that someone has successfully gained write access to the FTP site and is occupying significant disk space. This paper discusses the architecture enabling the creation, linking, and support of the monitors. The monitors may be running on the same or different machines and so appropriate communication links must be supported as well as regular status checks to ensure that monitors are still running. We will also discuss the construction of monitors for sensing the data, abstracting and characterizing data, synchronizing data from different sources, detecting patterns, and displaying the results.

Keywords: intrusion detection, architecture, sensors, distributed monitoring

1. INTRODUCTION

Intrusion into a system is a process rather than an event. There are multiple events that contribute to the intrusion. In general, the intrusion can be broken down into phases: probing the system, determining a way into the system, and finally exploiting the entry in some way. These phases may have multiple events that contribute to the overall goal of the intruder. The various events that contribute to the intrusion may be very different. As a result, the best ways to detect them may be different. Intrusion detection should not be thought of in terms of inspection of packets on the network but rather in terms of a coordinated effort to detect intrusion patterns across a variety of sensors looking for evidence among the many resources important to the computer system. Furthermore, the computer system should be thought of as the computers that together serve the user community rather than a single computer and the resources are all the functionality that enables the system to carry out its assigned tasks including the communications that exist among the computers.

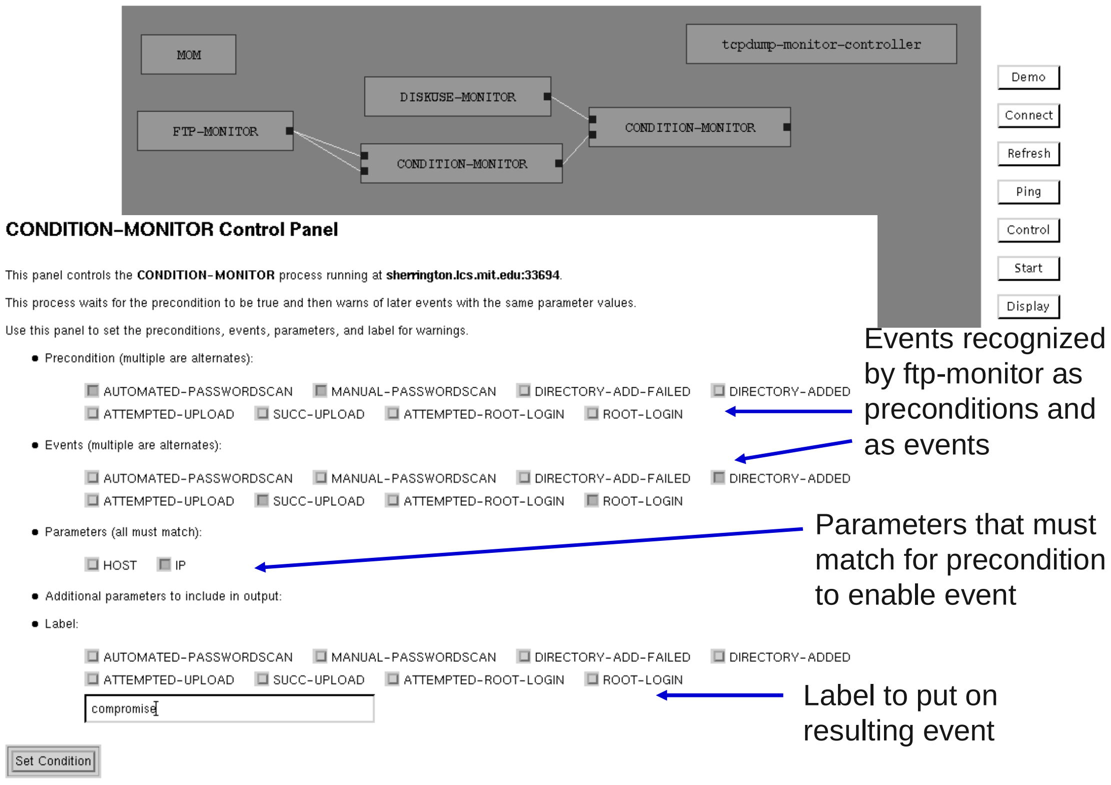
There are many sources of information about the functioning of the system from log files, to programs that test local and network resources, to the usual packet monitors. Each of these provides clues to the normal or abnormal functioning of the system and the possibility of a process of intrusion in progress. This project is developing the infrastructure necessary to support an integrated and flexible monitoring environment to use all of the available tools linked together in a coordinated distributed network such that the analysis can monitor the important aspects of the system and assist in the appropriate diagnosis.

2. AN EXAMPLE

The object in monitoring a system is not only to detect the act of intrusion but to detect any evidence of intrusion. Thus, we need to think in terms of the various activity in the system that might indicate unauthorized use of the system. For sensors we need tools that monitor kinds of system use such as disk use, ftp activity, email activity, logins, and so forth.

[*] wjl@mit.edu; phone 617-253-3508; fax 617-258-6762; http://medg.lcs.mit.edu/; MIT Laboratory for Computer Science, 200 Technology Square, Cambridge, MA, USA 02139

A representative kind of monitoring made possible by the MAITA architecture[1,2] is shown in Figure 1. This involves the use of an FTP monitor and a disk use monitor to detect an intrusion into the system. To enable these monitors to be used together it is necessary to have condition monitors to determine the sequence of events and to establish the necessary preconditions. The figure shows the monitor display in gray with the various monitors and connections and the HTML window with the control panel of the first condition monitor overlapping the monitor display. The monitor display shows the connections among the four monitors looking for the intrusion: an FTP monitor, a disk monitor, and two condition monitors. In addition it shows the MOM (monitor of monitors) and the monitor controller that enabled the user to create the rest of the monitors and run them on appropriate computers.


MOM
tcpdump-monitor-controller
DISKUSE-MONITOR
FTP-MONITOR
CONDITION-MONITOR
CONDITION-MONITOR
Demo
Connect
Refresh
Ping
Control
Start
Display
Events recognized by ftp-monitor as preconditions and as events
Parameters that must match for precondition to enable event
Label to put on resulting event
Set Condition

CONDITION–MONITOR Control Panel

This panel controls the **CONDITION–MONITOR** process running at **sherrington.lcs.mit.edu:33694**.

This process waits for the precondition to be true and then warns of later events with the same parameter values.

Use this panel to set the preconditions, events, parameters, and label for warnings.

- Precondition (multiple are alternates):

  [✓ AUTOMATED-PASSWORDSCAN] [✓ MANUAL-PASSWORDSCAN] DIRECTORY-ADD-FAILED DIRECTORY-ADDED
  ATTEMPTED-UPLOAD SUCC-UPLOAD ATTEMPTED-ROOT-LOGIN ROOT-LOGIN

- Events (multiple are alternates):

  AUTOMATED-PASSWORDSCAN MANUAL-PASSWORDSCAN DIRECTORY-ADD-FAILED [✓ DIRECTORY-ADDED]
  ATTEMPTED-UPLOAD [✓ SUCC-UPLOAD] ATTEMPTED-ROOT-LOGIN [✓ ROOT-LOGIN]

- Parameters (all must match):

  HOST [✓ IP]

- Additional parameters to include in output:
- Label:

  AUTOMATED-PASSWORDSCAN MANUAL-PASSWORDSCAN DIRECTORY-ADD-FAILED DIRECTORY-ADDED
  ATTEMPTED-UPLOAD SUCC-UPLOAD ATTEMPTED-ROOT-LOGIN ROOT-LOGIN

  compromise

Figure 1: Monitors and control for detecting FTP intrusion

The FTP monitor is running on the machine with the FTP server and continually analyzing the FTP log files to detect activity that might provide evidence of intrusion attempts. It utilizes two stages of pattern matching: classification of individual log entries and detection of patterns across entries. Three patterns applied to the entries are as follows:

- ((:user arg (user "anonymous") 331 GUEST-REQ :PASS ARG (pass) 230 GUEST-OK) anon-user 0)
- ((:user arg (user "root") 331 PASSREQ :PASS ARG () () 530 "Login incorrect.") bad-root-login 2)
- ((:user arg (user) 331 PASSREQ :PASS ARG () 530 "Login incorrect.") bad-login 1)

The first pattern detects the successful login of an anonymous user (classed "anon-user"), which is given a zero priority. The second is an unsuccessful root login and is classified as a "bad-root-login" with a high priority. The third is also an unsuccessful login "bad-login", but as a specific user with an intermediate priority and carries as an argument the name that was used, so multiple attempts as that user can be detected.

Detecting patterns across multiple entries include counts of events of a certain type, time delays between events, overlapping sessions, and the order of events. These can be illustrated by the following two patterns:

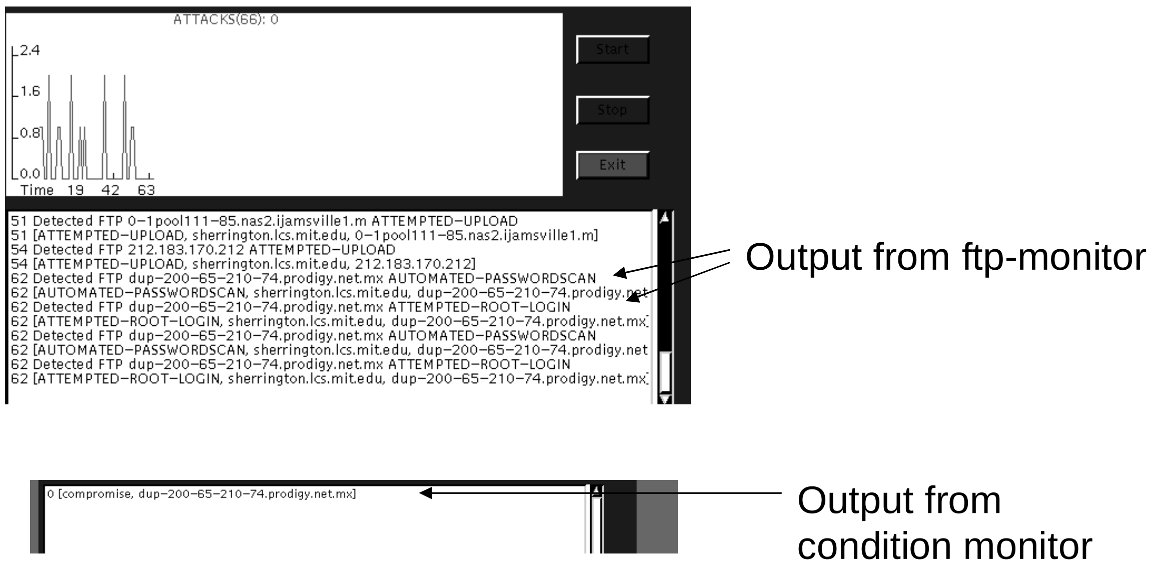
- ((and (> (src-count 'bad-login) 5)(> overlap 2)) automated-passwordscan)
- ((and (> (src-count 'bad-login) 5)(= overlap 0)) manual-passwordscan)

The first detects an automated password scan. The conditions required are at least five entries classified as “bad-login” and a timing condition that there be some overlap in the FTP sessions that they are part of. This condition implies that some automated process was doing the login attempts and not just a person manually typing at a computer.

Given an FTP monitor detecting events such as these, we can recognize part of the evidence for detecting an intrusion. The first problem is to detect some kind of password scan followed by activity from the same source that would normally only be done by an authorized user. If this were a well defined pattern, it could be another pattern in the FTP monitor. For most monitoring systems this would be another rule in the same formalism. However, since the definition may be evolving as we understand how intruders gain access to the system or may be dependent on the context or level of awareness desired on the part of the user, it is better to provide this functionality in a way that is open to modification whenever needed by the user. This is accomplished using a condition monitor that looks for the appropriate preconditions followed by a set of possible events along with any parameters they must share and then give the sequence a name that can be used by other monitors.

In this case both the precondition and the event come from the same FTP monitor. Once the condition monitor is connected to the two inputs, the condition monitor can query the FTP monitor to determine what kinds of events it can recognize and present these as options to the user connecting the monitors. The desired precondition is either kind of password scan. In the figure (the HTML window provided by the condition monitor) the desired events selected are directory-added, successful-upload, or root-login. These must be from the same IP address (parameters: IP selected) to trigger the condition and that condition is labeled a “compromise”, which becomes the event type available as output of the condition monitor for use by another monitor or a display.

In addition there is a disk use monitor that hourly checks the total disk space used on the file servers and local machine. This monitor provides as output hourly reports of the change for each disk, the fraction of total space used, and a warning when a limit is exceeded. This monitor provides input along with the first condition monitor to a second condition monitor. The precondition for the second condition monitor is the “compromise” recognized by the first condition monitor. Once this has happened, a warning of excess disk use from the disk use monitor triggers a condition of “probable unauthorized disk use”. The outputs are shown on the two displays in Figure 2.



```
51 Detected FTP 0-1pool111-85.nas2.ijamsville1.m ATTEMPTED-UPLOAD
51 [ATTEMPTED-UPLOAD, sherrington.lcs.mit.edu, 0-1pool111-85.nas2.ijamsville1.m]
54 Detected FTP 212.183.170.212 ATTEMPTED-UPLOAD
54 [ATTEMPTED-UPLOAD, sherrington.lcs.mit.edu, 212.183.170.212]
62 Detected FTP dup-200-65-210-74.prodigy.net.mx AUTOMATED-PASSWORDSCAN
62 [AUTOMATED-PASSWORDSCAN, sherrington.lcs.mit.edu, dup-200-65-210-74.prodigy.net
62 Detected FTP dup-200-65-210-74.prodigy.net.mx ATTEMPTED-ROOT-LOGIN
62 [ATTEMPTED-ROOT-LOGIN, sherrington.lcs.mit.edu, dup-200-65-210-74.prodigy.net.mx]
62 Detected FTP dup-200-65-210-74.prodigy.net.mx AUTOMATED-PASSWORDSCAN
62 [AUTOMATED-PASSWORDSCAN, sherrington.lcs.mit.edu, dup-200-65-210-74.prodigy.net
62 Detected FTP dup-200-65-210-74.prodigy.net.mx ATTEMPTED-ROOT-LOGIN
62 [ATTEMPTED-ROOT-LOGIN, sherrington.lcs.mit.edu, dup-200-65-210-74.prodigy.net.mx]
```

```
0 [compromise, dup-200-65-210-74.prodigy.net.mx]
```

Figure 2: Output from FTP and condition monitors

The first of these displays is connected to the output of the ftp-monitor and the second is connected to the output of the first condition monitor. Had there been a significant increase in the disk use, the final condition monitor would have signaled the “probable unauthorized disk use”. This set of monitors and displays illustrates a number of features of the MAITA monitoring system. First, the monitors can be assembled by the user in a flexible manner to meet the existing needs. For example, if the user needs more control over the kinds of abnormal disk use patterns that should be detected, an analysis monitor could be inserted between the existing disk use monitor and the condition monitor to apply the appropriate analysis to the quantitative data coming out the monitor and to generate classifications that more accurately reflect the abnormal patterns that might occur. Additionally, the user could substitute or add a disk backup monitor to the disk use monitor. The disk backup monitor analyses the daily backup logs and provides statistics on the volume of new files on each disk. This has the advantage of detecting all new files added to the system which means that use of the system as a transfer station or “file laundering” would be detected, whereas it might not be detected from statistics on increased volume on the disks. The disadvantage is that the new data is only available with the daily backups while disk use data can be generated as frequently as desired, although hourly seems to be a reasonable frequency.

The second feature is the ability to deal with data coming at a variety of time scales. In this example, the FTP monitor is detecting the difference between login failures that are overlapping in time because they are coming so rapidly and those that are strictly sequential. On the other hand, it is looking at changes in disk use from data that are generated once an hour or in the case of the backup monitor, once a day.

The third feature is the ability to put displays on any streams of data that are desired. Normally, one would just have the final unauthorized disk use warning displayed. However, if there is some indication that unusual FTP activity might take place, a display could be added to the output of the FTP monitor or some analysis of that output or any other place in the network of monitors.

3. MAITA ARCHITECTURE

To carry out a coordinated monitoring task with multiple sensors across multiple computers, we need an architecture to create the sensors and analysis tools on appropriate computers and connect them in a way that can be adjusted to the particular needs of the user. The architecture we have built is called MAITA, for Monitoring, Analysis, and Interpretation Tool Arsenal. The architecture supports a network of distributed monitoring processes that transfer data from one to another and communicate with each other to perform the monitoring tasks assigned by the human analyst.

The basic connectivity of a monitor is that it has a zero or more input connections from other monitors, zero or more output connections to other monitors, and a bidirectional communications port. There are several types of monitors, as determined by the number of monitor input and output streams:

1. Sensors: [no inputs, some outputs] these are monitors that sense some external source of information about the system and turn that information into a stream of data that can be used by other monitors. These function as the inputs to the network of monitors.

2. Displays: [some inputs, no outputs] these are the monitors that provide the results to the user. Their output is to the external world. Currently, there is a display that provides both line graphs of numeric data and textual notes (Fig. 4).

3. Transducers: [one input, one or more outputs] these are monitors that perform some analysis on the data, such as summing, filtering, or trend analysis. Usually there is a single output but if multiple functions are performed, there may be an output for each.

4. Combiners: [more than one input, one output] these monitors synchronize the inputs and perform some function that combines the inputs. The condition monitor discussed above is an example of this type of monitor.

5. Control: [no inputs, no outputs] these monitors function over the communications network performing services for the user and the other monitors. The primary example is the MOM or monitor of monitors that performs most of

the oversight functions for the network. Another example is a process for creating monitors, providing the user with a list of available monitors and forwarding the selected monitor specifications to the MOM for creation.

6. Control panels: These strictly speaking are not monitors, but rather the interface through the communications port to the monitor. Figure 1 shows two control panels overlapping. Most of the control panels are HTML forms to allow appropriate parameter setting and control. The control panel for the MOM is more sophisticated. It provides the display of the existing monitors and connections and provides commands to perform common operations, such as connecting, starting, and stopping monitors. It also provides access to the control panels for the individual monitors so the user has convenient access to all of the functionality. The control panel communicates with the MOM using commands formatted as HTML over a standard port. Using this protocol, it is possible to have more than one control panel. Indeed, it is often useful to have control panels on more than one machine to give different people access to the monitoring system.

All of the monitor terminals (inputs and outputs) provide persistent connections between monitors and operate over sockets. They provide buffering for the data to prevent loss of data when there are delays. There is a standard protocol for the serialization of the data and the data is transmitted with time stamps to enable appropriate synchronization. The actual format of the data is specific to the data type being transmitted, which is a property of the terminal and available via the communications link. A single output terminal can be connected to any number of input terminals. It simply delivers the same data to all of them. Thus, the only reason to have more than one output terminal is when there are very different kinds of data being produced. Even then, the same effect can be accomplished by connecting filters to the output to select the kind of data desired. Generating different kinds of data on the same terminal has the advantage that it is already synchronized so no additional work is necessary if more than one kind of data is useful in some later analysis or display. It is common to include both statistical data and textual messages on the same terminal so both can be used together in displays without recombining them. Connecting multiple monitors to the same output means that any output can be used as a display without interfering with other processing of the data.

The communications terminal operates using HTML protocol to provide one or two way connections between monitors on demand. This is the way that the MOM interacts with the monitors. There are a standard set of commands to this terminal to which all monitors respond and additional commands that are specific to the monitor. For example, all monitors respond to commands to start, stop, reset, identify themselves, quit, and generate a connection to another monitor. Many monitors have parameters to adjust their processing and these are set through specific commands.

4. SENSORS AND TRANSDUCERS

Sensors are the monitors that examine some external source of information about the system and turn that into a stream of data that can be used by other monitors and displays. Hence, sensors can be thought of as informational interfaces to the resources of the system. There are several types of sensors that we have developed as examples of the types of data available for monitoring:

1. Log file monitors: Most system processes keep log files that document the activity of one or more system resources. Examples for Unix systems include the system log, daemon log, and authorization log. The daemon log is where the FTP process records most of its activity (although some goes in the system log) and is the source for the data used by the ftp monitor described above. On many systems there are additional log files for processes installed on the system, such as the http log for the http server or the backup logs for a disk backup utility. The primary challenge for developing such monitors is parsing, categorizing, and summarizing the data in the logs.

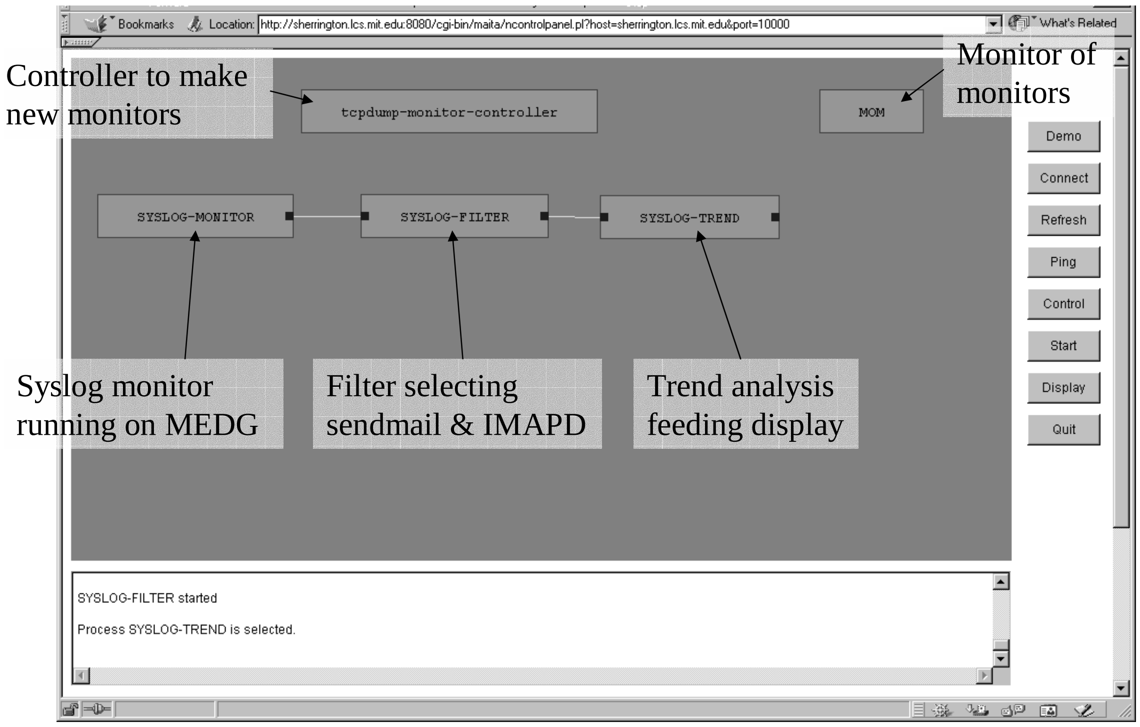
2. Utility monitors: Operating systems also provide a number of utility programs to determine the status of system resources. These can be turned into sensors by developing a monitor that runs the utility at an appropriate regular interval and summarizes the result. The diskuse monitor discussed above is an example of such a monitor. The monitor simply runs the “df”(disk free) utility and parses the output to determine how much space is available on each disk. Because much of the important information comes from the changes in disk use, the monitor maintains a data base of past information to provide appropriate tracking information. Other utilities include such programs as ping that can determine if specific hosts are up and available on the network. This is useful for keeping track of the status of a set of hosts that make up a system of importance to the analyst.

3. Interface monitors: Often a self-contained intrusion detector will be running on the system. This can be integrated into the system by providing a parser for the results of the intrusion monitoring that then produces a summary or analysis of the intrusion detector output. As an example we have designed a monitor that summarizes the output from snort[3]. This monitor keeps totals of the different kinds of suspicious packets encountered, totals of suspicious packets sent to specific machines and totals from specific hosts. This allows the monitoring system to look for higher level patterns not encoded in the snort rules.

The two most common kinds of transducers are filters and statistical time series analysis on the data. Filters simply select a subset of the incoming data for transmission to the next monitor. The advantage of having the filter as a separate monitor rather than incorporating that function in the sensor is increased flexibility. If different subsets are needed for different analysis paths, this can be accomplished with a single sensor feeding different filters. If the user wants to keep open the possibility of later analysis of other parts of the data stream (raised suspicion may suggest a search for enabling conditions), the user can send all of the output of the sensor to a database but only provide real-time analysis of a subset as specified to the filter.

There are many kinds of analysis possible on the data streams. We have transducers for various running averages and other statistics. In addition we have transducers that filter out "unusual" events, events that are beyond a given number of standard deviations from the running average. Such events are useful as early warning signals for possible intrusions – a way of doing anomaly detection with continuous data.

We also have a trend detector based on fitting line segments using linear regression and introducing a new line segment when the variance of a single segment is significantly worse than the variance when the segment is split into two contiguous segments. The parameter determining when the difference is significant can be set and determines whether the trends are short or long term.


Bookmarks
Location: http://sherrington.lcs.mit.edu:8080/cgi-bin/maita/ncontrolpanel.pl?host=sherrington.lcs.mit.edu&port=10000
What's Related
Controller to make new monitors
tcpdump-monitor-controller
MOM
Monitor of monitors
SYSLOG-MONITOR
SYSLOG-FILTER
SYSLOG-TREND
Demo
Connect
Refresh
Ping
Control
Start
Display
Quit
Syslog monitor running on MEDG
Filter selecting sendmail & IMAPD
Trend analysis feeding display

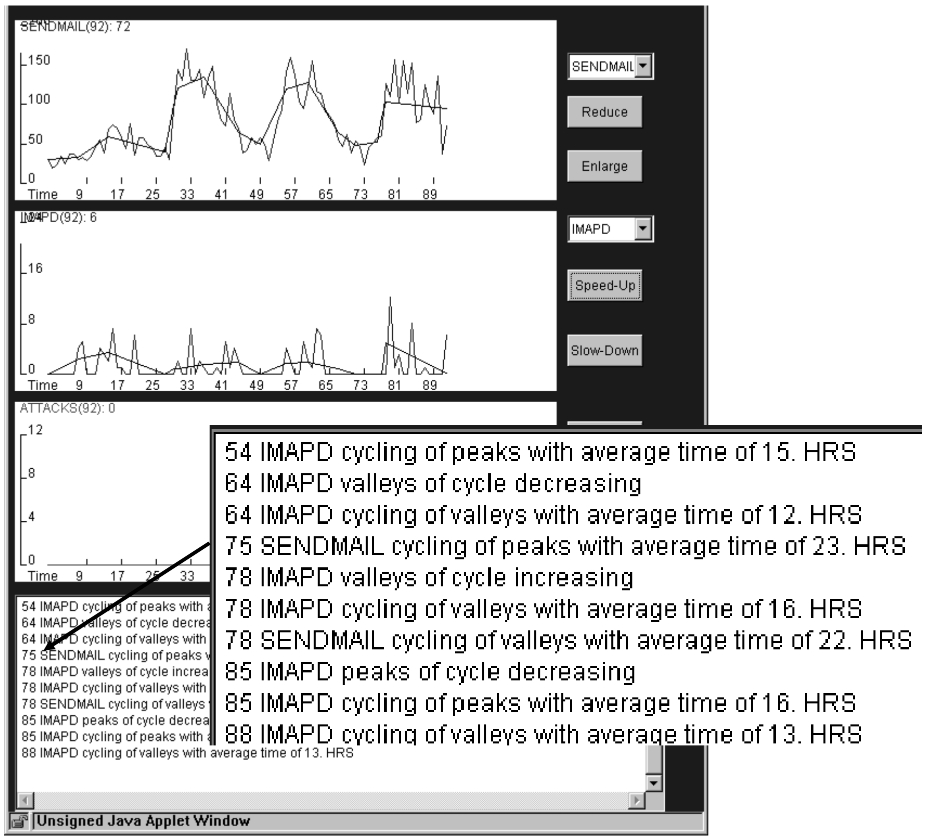
```
SYSLOG-FILTER started

Process SYSLOG-TREND is selected.
```

Figure 3: Setting up a monitor for sendmail and IMAPD

To make it easy to develop new sensors and other monitors, we provide a set of wrappers that provide the basic communications and interface functionality of the monitor. With these tools the developer specifies the structure of the monitor including input and output terminals and data types, any internal storage, and available parameters. The basic commands for the monitor are predefined although they can be superceded by functions specific to the monitor or a class of monitors. Any functions needed to carry out the processing desired must be written by the developer but the function that will be called when a packet of data is received on one of the input terminals is predetermined by the type of monitor and the name of the terminal. Appropriate macros are also predefined for binding the output terminals, creating appropriate packets, and transmitting the output. These functions also take care of the buffering necessary to ensure that no packets are lost. Whatever additional commands that might be needed for the monitor must be written by the developer but the infrastructure of handling the communications is provided. With these tools it is relatively easy to develop new sensors for resources available on a particular system, new transducers with analysis of interest to the users and other enhancements to the capabilities of the system.

To illustrate how the sensors and transducer would be used, consider a set of monitors shown in Figure 3. In Unix, the system log provides data on several of the important resources of the system. We have provided a system log monitor that parses the system log and keeps totals of the different kinds of data observed there.
Consider the situation where we are only interested in sendmail and IMAPD activity. We add a filter to the output of the syslog monitor and use the monitor control panel to select these types of records. Then the filter selects this part of the output and passes it to the output terminal. To this we connect a trend detector that fits line segments and does an analysis of their behavior. Finally, a display is connected to the output of the trend detector and the results are shown in Figure 4.



```
54 IMAPD cycling of peaks with average time of 15. HRS
64 IMAPD valleys of cycle decreasing
64 IMAPD cycling of valleys with average time of 12. HRS
75 SENDMAIL cycling of peaks with average time of 23. HRS
78 IMAPD valleys of cycle increasing
78 IMAPD cycling of valleys with average time of 16. HRS
78 SENDMAIL cycling of valleys with average time of 22. HRS
85 IMAPD peaks of cycle decreasing
85 IMAPD cycling of peaks with average time of 16. HRS
88 IMAPD cycling of valleys with average time of 13. HRS
```

Figure 4: Output of trend analysis of sendmail and IMAPD from syslog monitor

Notice that the trend analysis picks up the changes that take place over a few hours. This allows it to pick up cycling behavior over the course of a day. The trend analysis produces both the numeric data for plotting and the messages for use as categorical statements about the data for use in a condition monitor.

Plotting the trend data requires a display capable of plotting two streams of data on the same graph and also to handle data that revises previous data. Each time a new point is available, the trend analysis revises the slope of the last line segment, requiring it to be redrawn. When it is appropriate to break the line, the new point at the bend is back in time, requiring the display to find the appropriate place to make the change.

5. RELATED WORK

There are several types of work related to MAITA. There are many monitoring systems, as opposed to tool kits for assembling distributed monitors. These include commercial systems, research systems, and publicly available systems such as SNORT[3]. While these are intended to be complete systems for monitoring, they can also be viewed as components for MAITA as we have seen.

The system with an architecture that most resembles MAITA is the EMERALD intrusion detection system[4,5]. The EMERALD system provides a distributed set of monitoring processes, organized hierarchically to scale with the system being monitored. The EMERALD monitoring processes have a standard form that includes rules, statistical patterns, and a uniform method for each process to access the determinations of other processes. While the MAITA architecture provides similar capabilities, there is less structure enforced on the individual monitors and the ways that they can reason. As a result, the MAITA system allows more diversity in the kinds of monitoring processes that can be in the system. It is easy to develop simple monitors in MAITA while still allowing complex monitors to be developed using whatever technology is most appropriate.

There is also considerable work on monitoring knowledge and methods such as trend detection and “temporal abstraction”, especially in the work of Shahar[6] and Das[7] at Stanford. These efforts focus on representing temporal relationships and on methods for identifying patterns of temporal relationships. This work provides a good foundation for monitoring and indeed we intend to use some of the ideas in future work, but intelligent monitoring and analysis involves more than just temporal information. There is much reasoning that involves causal reasoning, association, correlation, and other types of relationships that must be supported as well to make maximum use of the clues available in the system. Our intent is to make use of as many kinds of data as are available.

The Guardian project[8,9] at Stanford has developed a very dynamic programming environment for the construction of flexible monitoring systems. The emphasis is on giving the system the ability to reason, during the monitoring process, about such things as the most appropriate data collection, interpretation and integration strategies. As a result little emphasis is placed on the ease of constructing simple monitors when those are appropriate. Thus, it is less flexible for creating tools that take advantages of knowledge of the system at hand.

Commercial technology for monitoring and control provides useful tools for some of the capabilities we seek, but tends not to have the flexibility to incorporate components outside of the particular product. The G2 system[10] offered by Gensym Corporation is an excellent example. This system provides many object-level monitoring capabilities, including the ability to accept inputs from several types of sources, a library of transducers (linear extrapolation, fourier transforms, etc.), and a knowledge-based reasoning component for constructing multisignal analysis systems. Unfortunately, G2 is structured as a standalone application and does not provide the capability of distributing the monitoring and display. As a single application it also requires the user to recompile and reinstall to add new capabilities to the system. Distributed and collaborative monitoring efforts instead require the ability to add new monitors to a running system to adapt to situations as they arise, as we have provided in the MAITA system.

6. CONCLUSIONS

We have developed and demonstrated an architecture for monitoring computer systems that provides the flexibility to adapt to the situation. It allows the user to assemble a set of monitors from sensors, transducers, combiners, and displays to provide the desired type of monitoring. It also allows the integration of existing monitoring tools from packet based tools such as snort to system utilities or special purpose monitoring programs into a system that combines their functionality into a common network allowing their analyses to be combined in meaningful and useful ways.

REFERENCES

1. J. Doyle, I. Kohane, W. Long, H. Shrobe, and P. Szolovits, "Agile Monitoring for Cyber Defense", Proc Second DARPA Information Survivability Conference and Exposition, 2001.

2. J. Doyle, I. Kohane,W. Long, H. Shrobe, and P. Szolovits, "Event recognition beyond signature and anomaly," in Proc 2001 IEEE Workshop on Information Assurance and Security, June 2001.

3. M. Roesch, "SNORT – Lightweight intrusion detection for networks," in Proc LISA 99: 13th Systems Administration Conference, November 1999.

4. P. A. Porras and P. G. Neumann, "Emerald: Event monitoring enabling responses to anomalous live disturbances," In Proceedings of the 20th National Information Systems Security Conference, Oct. 1997.

5. A. Valdes and K. Skinner, "Adaptive, model-based monitoring for cyber attack detection," In Recent Advances in Intrusion Detection (RAID 2000), Toulouse, France, Oct. 2000.

6. Y. Shahar, "A Knowledge-Based Method for Temporal Abstraction of Clinical Data," CS-TR-94-1529, Stanford University, 1994.

7. A.K. Das and M.A. Musen, "A comparison of the temporal expressiveness of three database query methods," in 19th Annual Symposium on Computer Applications in Medical Care, pages 331-337, New Orleans, LA, 1995.

8. B. Hayes-Roth, S. Uckun, J. E. Larsson, J. Drakopoulos, D. Gaba, J. Barr, and J. Chien, "Guardian: An experimental system for intelligent ICU monitoring," In Symposium on Computer Applications in Medical Care, Washington, DC, 1994.

9. B. Hayes-Roth, R. Washington, D. Ash, R. Hewett, A. Collinot, A. Vina, and A. Seiver, "Guardian: A prototype intelligent agent for intensive-care monitoring," Journal of AI in Medicine, 4:165-185, 1992.

10. R. Moore, P. Lindenfilzer, L. B. Hawkinson, and B. Matthews, "Process control with the g2 real-time expert system," In IEA/AIE '88: Proceedings of the First International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, volume 1, pages 492-497, Tullahoma, TN, June 1988. ACM.