Management, Administration and Security Aspects of Panoptes System

Saša Herceg*, Goran Protega*, Hrvoje Belani**
* KATE d.o.o. – Research & Development Unit, Unska 3, Zagreb, Croatia
Phone: +385-1-611-5251 E-mail: {sherceg, gprotega}@starcapital.net
** Fakultet elektrotehnike i računarstva, Unska 3, Zagreb, Hrvatska
Phone: +385-1-6129-745 E-mail: hrvoje.belani@fer.hr

Summary - This paper describes the System Administration, Recovery and Management (SARM) subsystem, which is used for establishing the interaction between the System and the system administrator. It provides information on the current state of the System and its subsystems as well as means for managing the System. Special attention has been paid to the issue of remote System administration, management and maintenance. The purpose of this document is also to address security problems in e-services dealing with mobile networks and to explain how these problems are resolved in the Panoptes system.

I. INTRODUCTION

Panoptes is a web-based application created by KATE Ltd. R&D team as part of eSchool concept. Panoptes enables the parents (or guardians) and children to receive information from school (notifications of homework assignments, general information, truancy notifications etc.).

This article focuses on the implementation of security, administration and system management functionalities in the application. These functionalities lie on the top of Microsoft .NET platform built on Panoptes. Our solution is a customization of platform tools that will comply with the requirements of the application.

Chapter II offers a brief overview of Enterprise Instrumentation framework – a Microsoft platform for administering .NET applications. Chapter III describes security, administration and management requirements placed on Panoptes. Chapter IV focuses on the implementation of these requirements and the resulting functionalities, and chapter V contains the conclusion.

II. AVAILABLE TECHNOLOGIES

Panoptes is built on top of Microsoft .NET Framework, requiring therefore the support on the server it is running on. This simplifies the implementation of system administration, because .NET framework provides the built-in API for numerous features of Microsoft Windows applicable in system administration.

One of the main issues in system administration is instrumenting the application for manageability in a production environment. Instrumentation is the act of incorporating code into the program that reveals application-specific data to a person monitoring that application. There are some important choices to be made regarding which information to convey, what data store to target and, if necessary, notify the right person.

For the purpose of instrumenting applications written on .NET platform Microsoft has developed the Enterprise Instrumentation framework (EIF). EIF is a comprehensive instrumentation application programming interface (API) especially targeted at multi-tier .NET, that provides a unified programming model for publishing application events. It leverages existing eventing, logging and tracing mechanisms built into Windows, including WMI (Windows Management Instrumentation), the Windows Event Log, and Windows Event Tracing.

EIF provides a flexible way of handling numerous application events, including exceptions. However, regarding exceptions, which need to be specifically handled, it is also possible to build custom logging and notification solutions.

EIF introduces event sources and event sinks that decouple events being raised from the underlying eventing, tracing or logging mechanism. In Enterprise Instrumentation, events are always raised from a specific event source. The event source configuration ultimately determines whether the event is raised or not, what information the event contains, and to which eventing, tracing, or logging mechanisms the event is routed. In all cases, information about the event source, such as event source ID, is passed as property values in the event object.

An event sink is responsible for taking an event and firing or logging it using a specific mechanism, such as WMI or Windows Event Log service.

WMI is the Microsoft implementation of an industry standard for accessing management information in an enterprise environment. The WMI event infrastructure provides the ability for applications to raise WMI events that can be monitored and handled. This approach decouples the notification rules and procedures from the code in the application. The application needs to raise only certain events that a monitoring system can catch and then implement the correct notification procedures.

The EIF configuration file, which is written in XML and is external to the application code allows the developer to
map connections between event sources and event sinks. This feature provides significant flexibility. It allows operations staff to specify what events will be raised, what data fields will be included, and which event sink(s) will receive the data.

III. FUNCTIONALITY REQUIREMENTS

Purpose of the SARM subsystem is to simplify the control over the operation and performance of the System. There are four major groups of functionality foreseen for SARM module:

- User administration – adding, editing and removing system users and content providers;
- System statistics – retrieving number of subscribers, messages sent etc. over a given time period;
- System diagnostics – reviewing system performance counters (processor, memory load etc.), retrieving log entries, running test procedures;
- System monitoring and alerting – registering important events and sending a notification to the appropriate person, if necessary.

The most important feature of the SARM subsystem lies in the monitoring functionality and it is the focal point of this article. Although Panoptes system doesn’t require constant human monitoring, system administration would be more efficient as well as cheaper if all critical events are instantly transmitted to the administrator and if the administrator has the ability to connect to the system from a remote location and the means for correcting the problem at hand.

Providing that Panoptes system is split into components, a reasonable thing to do is to develop a separate monitoring component that would receive information from them (errors, warnings, important event occurrences etc.) and undertake an appropriate action (i.e. process and forward the event to the right sink - receiver for the event). This component of SARM subsystem is called the Monitoring component.

Given the 24/7 prerequisite, a robust monitoring system had to be devised that could register both regular and irregular states of the application and notify the administrator. Many monitoring mechanisms are needed, which must include multithreaded processing of events that the components raise to signal that a specific state has been reached, with an exception of handling a process of irregular application states and some form of notification functionality to inform the system administrator or any other persons of the problems that have arisen. The events that the subsystem will monitor must include various system diagnostics and testing events and other various notifications that all the components could conceivably raise to inform the monitoring module of the regularity of their state. All critical exceptions that result from the application reaching an irregular state must result in an event being raised so that the monitoring module can process it and notify the administrator if needed. Since security issues will also be using the same notification mechanism, they will also be forwarded to the appropriate system maintenance personnel.

System administrator should be able to access all the essential administration and management functionalities from a remote location. Administering users (adding, editing, removing), monitoring system performance, reviewing system logs, testing the system, managing its components and analyzing statistics are among those offered to the system administrator from a remote location. Remote access is planned to have as little prerequisites as possible without threatening the security of the System. The Panoptes system gains more flexibility to accommodate different numbers of users if the system administrator is not bound by constraints of location, operating system, and installed programs. This requirement causes some security considerations.

In the development of different e-services, oriented to Web and mobile users that exist in heterogeneous environment, it is crucial to consider all possible security vulnerabilities. The result must be the secure system, with mechanisms to prevent security breaches and provide the reliable service to users. Security-relevant issues were considered in every component of the system, and secure through-tier and user-system communication is also assured.

In Panoptes system, potential security risks are considered on several different aspects:

- Web application security
- Mobile access security
- Data access security
- Secure communication.

Even though Microsoft leading technologies (Windows Server 2003, MS SQL Server 2000, MS .NET Framework, MS Internet Explorer 6, etc.) provide the users and developers with a great deal of security, the situation is still far from perfect. Just to illustrate recently discovered vulnerabilities of some technologies used in development of Panoptes system.

In particular case of MS Internet Explorer, valid SSL certificate (issued by VeriSign) can forge any other VeriSign SSL certificate. It is then used as intermediate certificate to allow 'secure' (HTTPS) access to the 'naughty' Web site [1].

Securing the application data is one of the most important security aspects. As some cases show, SQL Server 2000 was hit and caused damage by the Slammer worm [2]. After the world-known MSBlaster virus and patches for disabling it, it's expected that new security holes will be exploited soon [3].
Even the mobile phones aren't safe from viruses – malicious code attempting to do some kind of damage (e.g. frozen screen, automatically dial 911) [4]. Because the number of users is still rising, it will present great threat of overloading mobile networks.

The scope of this document does not allow detailed insight into all aspects of application security.

IV. IMPLEMENTATION AND RESULTS

Panoptes application is written in .NET languages, the server side supports .NET and the real question was whether or not to use the advantages of .NET framework, which hasn’t yet achieved global omnipresence. .NET provides developers with many signaling mechanisms. Panoptes application monitoring is based upon two of these mechanisms, namely upon exception management and event management. While none of these techniques are used extensively, their basic elements provide the building blocks for the system monitoring within the application. The Panoptes application is designed to be event-based, which is an appropriate architecture for a non-centralized, distributed application that is required to monitor and react to the changes in its’ status. The advantage of this approach is in the fact that the sender (all system components) and the receiver of an event (system monitoring module) are loosely coupled, which means that a sender-receiver relationship is created dynamically.

The Panoptes application’s monitoring module makes a logical distinction between affirmative and negative logs of the system’s activities. Event-raising mechanism is primarily used for affirmative logging, namely the components raise an event to inform the monitoring module that they have reached a certain regular state. These include “alive” events that the other components raise periodically, “trace” events that are used in system testing and maintenance, “security” events that convey information pertaining to some authorized or unauthorized access attempt to access the system, and many others.

On the other hand, exception management model is used to retrieve the information about the irregular states of the application. Both managed and unmanaged exceptions are caught within the code of all the components system. The exceptions are then wrapped into a special kind of an event, so that all the relevant information is preserved and sent to the monitoring module for event collection, processing and storage. The SARM subsystem acts as an event collector and provides information about the system as a whole by displaying it in an ASP page. Further analysis of the event data must be conducted by the system administrator. SARM subsystem does not monitor components’ individual logs, and should the administrator require this information, it must be accessed directly.

Having decided on .NET, the monitoring system, as the primary subsystem of the administration and management module, was implemented with these main functionalities:

- logging of various events raised by all components of the system. These are serialized and stored as XML files on the server machine with a defined lifecycle
- logging of both managed and unmanaged exceptions to the event log for easier storage and retrieval of information contained in the exceptions thrown
- monitoring received events through multiple watchdogs, which enables the system to autonomously notify the system administrator or any other person who should be informed about the problem and has the ability to rectify it.

Pseudo-code for event-raising mechanism in .NET C#:

- declare a delegate [5] and a client class that will have a method to receive the event [6] that matches the delegate’s signature
- register this method with the event to be a listener for the event
- create a class that inherits from System.EventArgs to encapsulate the event information and give it a suitable form.

A caller method that raises an event can be either synchronous or asynchronous. When a synchronous method is called, the caller is blocked until the method returns. In a multi-component application where components are very co-dependant it is unacceptable to have components raise an event and be put on hold until their event is being processed. On the other hand, when an asynchronous method is called, there is no result and the caller method proceeds without being blocked. Therefore, asynchronous calls act like messages, a one-way communication between a component and the monitoring module. Even though .NET has a set of library classes, which allow any method to be invoked asynchronously, the Panoptes monitoring module uses a different technique. Asynchronous behavior in event handling is achieved through the use of the system thread pool for processing the events that were raised synchronously. This ensures that the calling method returns immediately (when an item is queued for processing in the thread pool, the calling method returns immediately) and that all events that are received are processed at the fastest rate possible.

Figure 1 shows a typical example of event handling in Panoptes application. SARM subsystem consists of two major parts:

- Web interface - ASP pages used for access to the SARM’s functionalities
Monitoring component - provides the system monitoring functionality and collects and stores all the events raised by the system's components.

Monitoring component (or more specifically, SARM object) monitors other components of the system (Data Access Component (DAC), Interconnection and Comm and receives events defined in IMaintainable interface that they're implementing), the web user interface (UI) and system performance counters [7]. A security event in web UI is used an example of eventing because of its importance and universality.

In this case Panoptes web UI has detected a failed attempt of login to the system, caused by an error in username or password entered by the user.

If this is a third unsuccessful logon in the last minute (or any other specified time period), further logons from that location are inhibited. This situation could present a possible security issue that the administrator needs to know about. Username entered, IP address of computer from which the logon was attempted and standard description of unsuccessful logon (event type) are sent to the SARM object as parameters of a method call. This explicit call is due to the features of ASP pages and architecture of Panoptes system, other components of Panoptes system would wrap the information into an event, raise it and SARM would then handle the event in the same way.

Upon receiving the notification, SARM object creates a new thread in charge of processing it, and the thread invokes methods for publishing events. In this case the event needs to be urgently conveyed to the system.
System administrator can also access a variety of statistical information. System statistics can be retrieved in daily, monthly or yearly intervals for each school and type of message in the system.

Generally, when approaching security architecture design, it is advised to conform to certain patterns regarding security [9]: Single Access Point, Check Point, Roles, Session, Full View With Errors, Limited View and Secure Access Layer. According to the recommended security patterns, the security architecture has been developed.

Web application security covers building effective application-level authentication and authorization strategy [10]. With two types of roles defined, the access to the system features is role-delegated, supporting the user's actions inside the sessions. Authentication mechanism uses authentication cookies, and user credentials are securely stored in the database.

Because of using cookie authentication on user's Web access and Trusted Subsystem model to reach the application data, the secure communication is essential. It is provided between:
- Web server and Web browser (client),
- Web server and database server.

Encrypted communication is applied with the Secure Socket Layer (SSL). The clients use HTTPS protocol when accessing the login page. Use of HTTPS is optimized not to imperil the system performance.

V. CONCLUSION

Implementing system monitoring, logging and security is a necessary evil, often disliked by programmers. It requires programming time and its benefits usually remain foreseen until something goes wrong.

API’s such as EIF, as described in chapter II, are aimed at reducing the time needed to implement event handling, but for an application of Panoptes’s size, with only several event sources and sinks, customizing such mechanism would cost more time than developing own event collecting, processing and publishing procedures that better fit our requirements.

Component-based security approach that we implemented in Panoptes deals with security-relevant issues in every component of the system, but also assures secure through-tier and user-system communication.

VI. REFERENCES


