

Application Framework Design for Survivability

Hugh Matlock
Virtual Media Design, inc.
hmatlock@vmdi.com

Abstract

The Internet is a hostile place for software systems. Existing application development languages such as Java or C++ place a burden on implementers of survivable systems to insure that all possible attacks are detected and resisted. This paper presents a description of a framework-based approach to survivability that deflects some common attacks in addition to providing resource management and service recovery.

Request-Response Model

The system architecture is an Object Service Architecture (OSA), based on a set of distributed components whose actions are coordinated by message passing: request/response pairs. Requests are extracted from the appropriate message encoding and viewed as method calls on a specific object. Responses include a status code, return value, and possible data stream.

A web servlet component receives requests from the Internet (typically HTTP traffic) and instigates appropriate processing within the framework, returning the response data stream as a result page. While it may be protected behind a firewall, still there are several possible attacks that need to be recognized and prepared for. This is the job of the framework.

Denial of Service Attacks

To continue to provide service to legitimate users while under a Denial of Service (DoS) attack, a software system must winnow the relatively few valid requests from among the much more numerous hostile requests. In essence, valid requests are to be given a higher priority for processing, and invalid requests must be discarded if they become too numerous. The task is one of efficient discrimination.

Incoming requests can be divided into three classes:

1. requests that are part of a current session
2. valid requests to start a session
3. everything else

Requests in class 1 are delivered to the message format parsers for regular processing (see below). The discrimination between classes 2 and 3, determination of whether a request is a valid request to start a session, could be a lengthy process. It can involve authenticating the user, which might mean a database query to check a password. This process is broken down into a series of stages, filtering invalid requests using a minimum of resources at each stage.

Requests that require further checks are queued while the input request queue is scanned for class 1 requests. If queues reach a certain maximum size, requests are discarded. This provides for graceful degradation of service in the face of attack.

Syntax-Directed Attacks

A typical syntax-directed attack presents invalid or unusual parameter values in an effort to cause the system under attack to fail. [1]

The framework erects a barrier between the incoming transaction and the application code. An interface definition defines the valid object methods for each class of target objects. Method parameters are described using XSchema data types [2]. In addition to defining the base value type (integer, string, etc.) these data types permit the definition of minimum and maximum values, string patterns, and the like. The ability to bounds-check parameter values provides protection against attacks based on invalid parameters.

While this approach can be adapted to any API, it is perhaps most suitable to interfaces such as SOAP, that already use XML for information transfer. In this case (which may be expected to be common as we move forward) the requisite XSchema may already be available.

Obviously, the data type validators must be carefully implemented, so as not to fail themselves. However, each parameter data type need have only one implementation, making the task smaller and less open-ended than the alternative: insuring that each parameter on each API is properly checked. This approach also has the virtue that it can provide some protection against possible vulnerabilities in third party libraries.

In addition, the message format parsers that are specific to the incoming protocol syntax must correctly extract object references, method names, and parameters. With the increasing reliance on standard message formats (e.g. SOAP, IIOP, BizTalk) comes the hope that a small number of message parsers will suffice for many applications.

Data Model and Resource Management

The Java Data Objects (JDO) model is used for access to persistent store. In this model, object references cause a "load on demand" procedure to run if the objects are not currently in memory. Access rights are checked during the load procedure. This provides a greatly enhanced degree of robustness, as the load procedure can take corrective action if errors occur.

After processing, storage is based on synchronizing the persistent store with the in-memory image. At this time, access rights and disk resource quotas can be enforced systematically. The synchronization is performed as a transaction, maintaining data integrity in the event of failure.

Service Recovery

Each component of a distributed system may depend on others for proper operation. Access to these components can fail for many reasons, not only attack. When access to an external service fails, a recovery process may attempt to restore access. The on-demand loading and synchronization-based storing procedures can detect errors and perform certain sorts of recovery operations.

In fact, the entire sequence of events, from database creation, through table creation, and table content can be performed if necessary, based on declarative information in a configuration database. This provides an ability to continue transparently in the event of a loss of a write only database (such as a logging database). It also provides the ability to continue in a degraded mode (such as with a read-only backup) in the event of the loss of a writable database.

Notes

[1] Rauli Kaksonen, Marko Laakso, and Ari Takanen, *Vulnerability Analysis of Software Through Syntax Testing*
<http://www.ee.oulu.fi/research/ouspg/protos/analysis/WP2000-robustness/index.html>

[2] Paul V. Biron and Ashok Malhotra, *XML Schema Part 2: Datatypes*
<http://www.w3.org/TR/xmlschema-2/>