4-29-2010

Providing Availability on the Poly^2 Framework

Ankur Chakraborty
Purdue University - Main Campus, ankur.chakraborty@gmail.com

Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Ankur Chakraborty

Entitled
Providing Availability on the Poly^2 framework

For the degree of Master of Science

Is approved by the final examining committee:

Eugene Spafford Chair	
Melissa Dark	
Dongyan Xu	

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Eugene Spafford

Melissa Dark

Approved by: Eugene Spafford Head of the Graduate Program 04/28/2010 Date

Graduate School Form 20
(Revised 1/10)

PURDUE UNIVERSITY
GRADUATE SCHOOL

Research Integrity and Copyright Disclaimer

Title of Thesis/Dissertation:

Providing Availability on the Poly^2 Framework

For the degree of Master of Science

I certify that in the preparation of this thesis, I have observed the provisions of *Purdue University Teaching, Research, and Outreach Policy on Research Misconduct (VIII.3.1)*, October 1, 2008.*

Further, I certify that this work is free of plagiarism and all materials appearing in this thesis/dissertation have been properly quoted and attributed.

I certify that all copyrighted material incorporated into this thesis/dissertation is in compliance with the United States' copyright law and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless Purdue University from any and all claims that may be asserted or that may arise from any copyright violation.

Ankur Chakraborty
Printed Name and Signature of Candidate

04/28/2010
Date (month/day/year)

*Located at http://www.purdue.edu/policies/pages/teach_res_outreach/viii_3_1.html

PROVIDING AVAILABILITY ON THE POLY^2 FRAMEWORK

A Thesis

Submitted to the Faculty

of

Purdue University

by

Ankur Chakraborty

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

May 2010

Purdue University

West Lafayette, Indiana

To my parents who gave constant encouragement and kept me going through with their good humour, to Arjun for reminding of the encouragement and the humour and to Dr. Anish Mathuria for shoving me headfirst into the pool of research.

ACKNOWLEDGMENTS

The author would like to thank Dr. Eugene Spafford, Dr. Melissa Dark and Dr. Dongyan Xu for the guidance and advice provided by them for the course of the research done for this thesis.

The author would also like to thank Keith Watson for the help and support provided during the course of this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Chakraborty, Ankur. M.S., Purdue University, May, 2010. Providing Availability on the Poly^2 Framework. Major Professor: Eugene Spafford, Melissa Dark.

Availability is not often a primary concern for frameworks meant to provide security. Poly^2 is one such framework. It provides us with a hardened foundation based on secure design principles to run mission-critical services. While, the primary focus of Poly^2 till now seems to have been fault isolation, we will now attempt to add recovery as well. However, current techniques may compromise the security principles on which the framework was originally built. We propose a hybrid system based on two popular techniques to rectify the same.

CHAPTER 1. INTRODUCTION

The Poly^2 (short for poly-computer poly-network) paradigm (Bryant et al., 2003) posited a hardened framework in which the critical network services of an organization can operate. This framework was intended to provide robust protection against the attacks running within this domain. As a part of this approach, network services are separated. They are placed in application-specific systems and use minimized operating systems. This allows the isolation of untrusted systems and services. This also prevents vulnerabilities in one system from affecting other systems. The core design problems were inspired by Saltzer and Schroeder principles (Saltzer & Schroeder, 1975) and Neuman's principles (Neumann, 2000).

While the paradigm looks at securing the system and from containing any unwanted breaches, it however leaves the issue of availability open. The setup inspired by the Poly^2 ensures security and isolation of services to prevent the repercussions of breaches from spreading beyond the affected service. Availability has been defined in Bryant et al. (2003) as systems that are continuously operational for long periods of time.

1.1. Definitions

The challenge with working on availability in this system is that the security aspect of the paradigm will have to be kept at the forefront. This, of course, would mean that a definition of availability has to be adopted that would fit into this paradigm, which is why the above definition may seem inadequate.

The notion of availability has been borrowed from the telecommunications industry and the definitions also seem to have followed those norms. These norms have laid more stress on fault tolerance due to it's inherent understanding in the matter of telecommunications. By this, availability as defined by the Alliance for Telecommunications Industry Solutions Telecom Glossary (2007) is "The ratio of (a) the total time a functional unit is capable of being used during a given interval to (b) the length of the interval". In Ross (1997), availability is given as the probability that a system is functioning at time t. In Laprie (1992), availability is given as readiness for correct service.

However, all the above definitions relate to dependability only. Availability is very closely related to security as security breaches have often led to availability failures, examples of this being denial of service attacks, which often lead to the service being unavailable. From a perspective of security, the International Standards Organization (2000) defines availability as "Ensuring that authorized users have access to information and associated assets when required". However, as is asserted in Rosseba et al. (2006), availability has to be a consensus between dependability, security and usability. Therefore, the definition that is given in International Standards Organization (1989) and the International Standards Organization (2001) seems the most suited.

Hence, the above definition that says, “Availability is the property of being accessible and usable on demand by an authorized entity” will be adopted for the purpose of this study. With this definition in mind, the research question will be stated.

1.2. Research Question

How can services under the Poly^2 framework be made accessible and usable on demand by an authorized entity through fault tolerance mechanisms? What would be the costs involved in such a feature and how would they be comparable to the alternatives?

1.3. Scope

There would be certain constraints that would be placed upon the problem. The study would be attempting to provide fault tolerance within the context of the framework. However, there may be an issue where a better availability solution may have to be compromised to stay within the original principles of the framework.

The scope of the problem is limited to fault recovery. As a result, the researcher would be studied implementation of fault tolerance techniques in the context of providing secure services. However, Fault correction or prevention of vulnerabilities are beyond the scope. When a service has been compromised, the exact same service can be restored from a backup replica. This could potentially mean a repeat of the event but this is beyond the scope of the current endeavor.

1.4. Significance

The question of availability arises when the system is required to be continuously operational for long periods of time. The Poly^2 framework is meant to contain within itself the services that are mission critical to an organization (Bryant et al., 2003). Due to this nature of framework, it is necessary that proper measures be taken to ensure the protection and continuation of these services.

With the changing focus of security research, aspects other than security are now also seen to be addressed (Meadows, 1994). Due to the prevalence of denial-of-service attacks, it has become more important to address the issues of availability in secure systems as well. Access control lists can be seen to address the issues but unfortunately while they can prevent other processes from denying the service access to resources, they cannot guarantee the actual provision of the service.

It has been acknowledged in Kyamakya, et al. (2000) that totally secure systems are not practical and that measures need to be taken to ensure that the flow of services and information is uninterrupted. It has been mentioned that the system requirements need the essential and non-essential services to be separated. The essential or mission critical services need to be made accessible and usable by authorized entities.

The issue of integrating the paradigms of security and fault tolerance has been explored as a part of the literature review. However, this issue is seen in the real world as two separate ones. But ensuring that secured services remain accessible and usable is necessary for the organization to function since businesses are becoming more and more dependant on computers and computer networks. These services although often protected may still remain vulnerable to faults such as hardware failure, excessive load, etc. Added to this, it also remains that all possible exploits cannot be accounted for within these services. It has to be ensured that these services are available in the presence of any possible faults (malicious or otherwise). While stopping them may involve techniques that could be more expensive, the Poly^2 approach has been the isolation and absorption of these faults, the colloquial “rolling with the punches”. This is what makes the issue of providing this aspect of availability within Poly^2 significant.

However, there are a number of approaches to providing reliability aspects to the Poly^2 framework. We will be using the cost-benefit analysis based approaches to look at the technique proposed by the author vis-à-vis the existing ones. This would give a more clear indication as to the benefits posed by the technique.

1.5. <u>Limitations</u>

The following were the limitations under which the research was carried out:

- While this study is to determine availability techniques for secure network frameworks, the researcher will be focusing on the Poly^2 framework.
- For the cost-benefit analysis, the researcher will be focusing on replication-based techniques.
- For the purpose of the implementation, the hardware of implementation will be limited to the currently deployed hardware for Poly^2.

1.6. <u>Delimitations</u>

The following would be the delimitations under which the research would be carried out:

- For the purposes of the project, the replicas will also have the same vulnerabilities as the original services.
- The standing of the cost-benefit analysis is restricted to the service providers.
- The cost-benefit analysis will be limited in its extent on the cost to the Poly^2 framework.

- The cost and the benefits would be applicable to the service providers whose services would be falling under the purview of Poly^2.

1.7. Chapter Conclusion

This chapter introduced the research the research being carried out for the thesis. Also explored was the significance of the research as well as the constraints within which it is being carried out.

CHAPTER 2. LITERATURE SURVEY

There are primarily four components to a system providing availability – fault detection, fault-tolerance, fault-recovery and fault prevention. The researcher's concerns are limited to fault-tolerance. This chapter will begin by looking at existing literature in fault-detection since to be able to be tolerant to faults, it is essential that they be detected first. It will then be looking at various techniques for providing availability through replication and other techniques. This section will be concluding with techniques that would harmonize both availability and security.

2.1. Fault Detection

For failure detection, absence of proof of aliveness would be taken as evidence of the same. Heartbeat messages provide perfect failure detection (Anderson & Lee, 1982). They are exchanged between replicas during error-free periods to keep a mutual track of redundant systems within the infrastructure. Four accelerated heartbeat protocols were suggested to be used between replicas (Gouda & McGuire, 1998). These protocols were based on the principles of low beat rate, low detection delay and low probability of premature termination. Low beat rate is the rate at which the beat messages are sent, the detection delay is the longest period that can be allowed to pass after one process terminates and the probability of premature termination is the probability that the heartbeat protocol declares termination due to the loss of a beat message. These were with the purpose of reducing overhead and increasing the responsiveness and reliability of the protocol, respectively. The first one is a binary protocol involving only two processes, the next one is a static one with n processes, and the third and the fourth protocols start with one process and gradually expand to include more. The first three are static in the matter that they do not allow members to leave (the first two do not allow members to enter either). The fourth one allows members to leave the heartbeat network as well. Later, a multi-level heartbeat protocol architecture is provided that is primarily to prevent single points of failure (Li et al., 2009). Heartbeat protocols are primarily deployed as processes on the application layer. This brings up the restrictions of the scheduler on the protocol that might result in the protocol being inadvertently compromised. It was proposed to integrate the protocol with the network layer (Wang & Li, 2008). This technique let the protocol run directly from the kernel. However, it required that a separate network interface be used that would require major changes in infrastructure.

Often, perfect failures (which are detected by the heartbeat protocol) may need to be need to be achieved in cases such as network breaches where failure is not evident but may need to be declared to prevent further damage and to let recovery take place. For this purpose, watchdog timers or a STONITH like approach, which uses a network power switch to reboot the service, may be used. STONITH means “Shoot The Other Node In The Head”.

2.2. Replication

Due to the presence of single services running on each of the nodes on the application nodes, a replication approach may be seen as a way towards achieving availability. The semi-active replication (also known as leader/follower) approach (Schmidt & O’Ryan, 2000) is one in which each non-deterministic action is carried out in the replicas and then the leader notifies them that it has been successfully carried out to keep the states consistent. This approach seems to have limited applications. It requires that the files remain read-only. Due to large volumes of data transfer, this approach may seem unsuitable for services requiring high throughput. In the work by Deplance et al. (1999), a semi-active replication strategy is implemented on the Chorus/ClassiX distributed operating system. The main features of this system were that the management of the replicas was transparent to the application designer and the application designer had only to describe the distribution and replication of actors.

Active replication on the other hand has all nodes concurrently process the incoming traffic (Wang et al., 2001) with the clients seeing only one server that is responding. There are two approaches to implementing this. The first one is that all the nodes use reliable multicast protocols to deliver the traffic to them such as those proposed in Deering (1989). This technique, however, requires that both the client and the server are reliable multi-cast aware i.e. they are able to use Application Programming Interfaces (APIs) for atomic broadcasts/multicasts. Another way as proposed in Ayari et al. (2008a) and Ayari et al. (2008b) is to provide a proxy or gateway as an intermediary to deliver the messages to all the replicas. This may lead to higher processing overheads and possible changes in the hardware architecture. Moreover, it ends up adding another single point of failure. If either the proxy or gateway were to fail, it would lead a nullification of the replication scheme. Architectural modification may also be needed to accommodate the additional intermediary. Also, another suggested technique of active state replication reduces the dependence on a single point of failure. This involves the replicas passively intercepting and processing the traffic offered to the primary server. However, only the primary server responds to the traffic. This approach requires that the replicas be equivalent to the primary server in regard to hardware and software capabilities. This seems suited to the Poly^2 architecture where the replicas of the individual service servers would be ones which would have exactly the same hardware and configurations. However, this approach might lead to a breach of security. Because the primary server's state is replicated on all the replicas, a breach would also be replicated leading possibly to recovery not occurring and hence defeating the whole purpose of replication. In Engelmann (2007), approaches of replication to High-Performance Computing (HPC) are described. An attempt is made to provide a generic programming interface to create replication environments for different services rather than building them from ground up. They provide three distinct models for replication. An Active/Standby model is one in which there is one active service and atleast one standby service that operates either in warm or hot standby mode. An asymmetric active/active model is one in which there are two or more active services, each of which operates independently and share no state, however they do have their own standby services sharing states with the either of the active services. A symmetric active/active model is one in which there are two or more active services which have the same capabilities and share the same global state.

2.3. Other Availability Techniques

A checkpointing approach (Laadan et al., 2005) can be seen as a passive replication technique. The server state is periodically copied to a standby server and restored from the checkpoints during failures. This approach works at the transport layer in a protocol independent manner. This approach decouples the operating system from distributed application. The application itself is run on a virtualization layer. This allows the checkpointing application to simultaneously and smoothly checkpoint the distributed application. The main problems seem to be lack of perennial consistency between the primary servers and the replicas and the significant overheads that may be incurred when creating the checkpoint. From a security perspective, however, this approach may allow a fallback to a point where the system was not compromised.

In Bhide (1992), availability schemes are discussed with two very distinct scenarios. One is with a Shared Nothing Database with Asynchronous Replica Management and the other is with a Network File System. In these schemes, Bhide discusses a basic asynchronous replication technique on which both the replication methods are based. All the transactions are carried out on the primary entity and the transactions are located on a log that is often kept in a separate domain to protect it from failure. The log is used to run the transactions on replicas. This update can be done either upon failure or be carried out regularly. This is seen to require large amounts of network bandwidth but is efficient in terms CPU and I/O required. With the NFS, there are two layers of replication that are carried out. One is at disk level and the other is at server-level. Each server services two similar disks on which the data is replicated so in case of disk failure on one of the disks, the other disk can maintain the data. With the servers, in case of failure, the replica server can obtain the current “duplication cache” from one of the disks and carry on the service keeping the failure and recovery transparent to the users. However, in this case the protocol itself is stateless and the effects of this technique with stateful protocols are yet to be seen. In case of the asynchronous replica management, the transaction state is used to maintain the replicas. All changes are recorded onto a log and in the event of a failure, the replica fetches this log and updates it’s state to the primary server’s point of failure.

All these techniques, however, give security a secondary consideration. They do form the basis and are also the most commonly used techniques used to provide availability. We will now look at some of the existing work at harmonizing fault tolerance and security.

2.4. Security and Fault Tolerance

In Meadows (1995), it is argued that the current security paradigm is a subset of the types of approaches used in dependability. This is to move away from the currently used approach of a worst case scenario. Instead in Meadows (1994), it is suggested moving towards a failure model. Among the approaches missed, is the particular one of fault tolerance. Though Meadows argues that fault tolerance principles have been subconsciously used in security paradigms such as operating systems that have multilevel security to keep possibly malicious code in the untrusted portion of the system. Taking this view, Meadows builds a fault model for security based on the three areas of faults in security mechanisms, hostile attacks on the system and compromises due to poor usage of the security mechanisms. However, we shall be taking an approach which goes in the opposite direction in which Meadows has been pointing. Instead of fashioning the security paradigm around the dependability paradigm, we will be attempting to fashion fault tolerance around the security paradigm.

In Price (2001), the author attempts to provide fault tolerance to security protocols. For this purpose, he redefines failure in case of protocols to say that it occurs when the goals of the protocol specification are not met. He gave a fault tolerance mechanism based on anonymity to provide security against denial of service attacks. Next, he attempted to show the effect of state on fault tolerance in secure services. He gave, on the basis of statelessness, two notions of replication for secure services. A stateless server involves complete independence between the replicas to prevent a malicious server from exposing the rest of the replication group. However, this form of disconnection is not feasible for providing availability. A semi-stateless server, on the other hand shares a limited set of information with other servers within the group. This allows a form of limited replication that can be controlled through the security policy. This also prevents the malicious server from exposing the group completely. This technique is one that can be easily integrated into a hardened framework where security policy on the rest of the network is determined. However, unlike stateful replication, this may lead to loss of functionality and may defeat the very purpose of replication if the security policy is not configured properly.

In Malkhi and Reiter (1998), a secure replication technique is suggested based on a quorum. In this case, there are a large number of servers and the operations are carried out on a quorum of the same. i.e. if there are a total of n servers, the operations are based on the responses from a quorum which is typically $\sqrt{n}$. This allows the systems to tolerate a wide range of failures. This was primarily meant to service applications such as Public Key infrastructures, Publishing and dissemination and National Voting Systems.

In Cachin and Poritz (2002), a replication technique that maybe tolerant to intrusions is suggested. It involves using a cryptographic channel for the purpose of atomic broadcasts in order to securely maintain the causality of the system. It is built upon multiple layers of cryptographic primitives and binary agreements upon which the replication is carried out. The atomic broadcast itself is considered a channel for the purposes of this system through which a secure fault tolerance system is built.

In Schneider and Zhou (2005), a form of distributed trust is posited. This involves basing the fault tolerance on the distributed trust. This way, if a single or a small number of components are disabled or compromised, the ensemble of state machines would still outvote their actions. The approach suggested here however does tend to go against traditional replication paradigms. There suggestions are to use multiple server configurations that may not necessarily have common software. While, this approach will prevent vulnerabilities from being replicated. It will also create issues with coordination of replicas.

In Reiter (1995), a toolkit is provided to attempt to provide a replication solution to security-critical services. Called Rampart, it attempts to provide correct and reliable replication service inspite of malicious intent of attackers attempting to enter the system. This toolkit takes the approach of relieving the developer of issues regarding replication and takes care of the aspect. The primary components of the toolkit are the secure atomic multicast, reliable multicast and the group membership protocol. Cryptographic attributes were used to insure that the protocols functioned in asynchronous environments.

2.5. Looking at Cost Benefit Analysis Techniques

Cost benefit analysis is used to allocate resources more efficiently to certain projects. Cost benefit analysis are performed either during the course of the project, known as media res; while the project is being considered, known as ex ante; or at the end of the project, also known as ex post (Boardman et al, 2006). Cost Benefit analysis is often used to show the superiority of a project with respect to alternatives.

In Boardman et al, (2006), a fundamental approach to cost benefit analysis is examined. It starts with the very basic process of identifying the set of alternative projects that may be used instead of the one under consideration. Next, the actor who plays the principal role in the project is identified. This actor is said to have standing, i.e. it is their costs and benefits that are being studied. Next the impact of the project alternatives is studied as benefits or costs. Along with this, the measurement criterion is also specified. Next the impact of the project is specified over its lifetime. This impact is monetized and the net value of the project is discovered.

The above method deals on a generic level of cost-benefit analysis with non-information technology projects as well. However, in cost-benefit analysis of IT products, it is often difficult to evaluate the benefits (Sassone, 1998). There are some methodologies that are discussed to carry out cost benefit analysis on Information Technology products. Decision Analysis provides an operations research perspective to making choices. It is particularly useful for evaluating systems meant to support routine decision making. A structural model represents a line of business or function and the impact of the information system on the costs and benefits of the function. A breakeven analysis is carried out by parametrizing the benefits obtained. This is often carried out when the benefits cannot be quantified. Subjective analysis is carried out again when the benefits are intangible, speculative or uncertain. It is often carried out in response to a fixed cost.

However, for information security projects a different approach to cost benefit analysis is required. The benefits are often tied directly to the costs making cutting costs an important benefit (Gordon and Loeb, 2005).The magnitude of cybersecurity breaches often defines the cost and whether it is direct or indirect cost or whether it is implicit or explicit. The goal with implementing any project has to be the maximization of benefits with respect to the costs.

The benefits with respect to cybersecurity are often cost avoidance; i.e. avoiding the costs that occur due to cybersecurity breaches. There are a number of models that can be associated with the treatment of cost avoidance as possible benefits in cybersecurity. Often, it is required that the anticipated benefits be compared to costs over time. A model that is associated with such an approach is the net present value (NPV) model. NPV refers to the difference between the present cost of the project and its initial cost. This approach is useful in considering incremental investment to cybersecurity. A positive NPV leads to an acceptance of the measures, while a negative or zero NPV leads to rejection or indifference. The NPV shows how much the anticipated benefits would exceed the present value of the anticipated costs. The internal rate of return (IRR) model is a derivative of the NPV model. Instead of dealing with present value, it uses the average cost of capital as a metric. The cost of capital is the minimum rate a project needs to earn in order to keep the organization’s value from reducing. The IRR is the discount that makes the NPV equal to zero. An IRR greater than the average cost of capital leads to an acceptance of the measures, while lesser or equal IRR would lead to rejection or indifference.

The above, however, have dealt with how to carry out a cost-benefit analysis. Wei et al (2001) carry out the cost-benefit analysis of installing a Network Intrusion Detection System (NIDS). They use the Annual Loss Expectancy (ALE) as their metric to determine the cost involved. ALE here refers to the product of the number of incidents with the total number of incidents that have occurred. They use a cost model based on operating costs, damage costs and response costs where operating costs are the costs involved in analyzing the stream of traffic, the response costs involve the costs of responding to the intrusion while the damage costs involve the costs that might be incurred if the intrusion is successful. The cost benefit is carried out based on the cost incurred due to the ALE, the costs of the countermeasure and the savings incurred due to the same.

2.6. Chapter Conclusion

This chapter explored the various fault detection techniques, replication methods and other availability models in the course of this chapter. It then looked at the various attempts that have been made at marrying the concepts of security and fault tolerance. To get an indication of the benefits accrued due to the project, a cost benefit analysis is carried out. This chapter looked at some approaches that have been explored in the cybersecurity domain. Keeping these in mind, the approach towards implementing availability within the Poly^2 framework will now be looked at.

CHAPTER 3. METHODOLOGY

3.1. Implementation

There would be two aspects of looking at the problem. One would be an analysis of the costs associated with implementation of the aspect. The second would be the implementation itself.

The research focuses on carrying out a study of high availability techniques. For the analysis, a cost-benefit analysis of the techniques we have suggested with respect to the Poly^2 system was carried out. This analysis took into account changes to software, hardware, configuration as well as architectural changes. Potential benefits will be including fault tolerant characteristics in the Poly^2 framework. The Poly^2 framework is based on secure design principles. As a part of the analysis, it has to be taken into account as well as how much would these design principles be affected. Upon the completion of the analysis, the most viable of the techniques will be implemented.

The nodes of the Poly^2 framework are single service servers. Each of them has customized operating systems and minimized network stacks. The operating system is tuned to support a single, specific application. This tuning is carried out primarily in terms of performance and security. This specialization allows detection of deviations from normal behavior. With respect to the analysis, it has to be seen as to how these techniques will affect this configuration and to what extent.

The researcher has two possible approaches to the problem posited. He can either adopt an active replication technique, a checkpointing approach or a hybrid of both. An active replication technique would seem suited to the purpose due the lack of service consolidation and the presence of a gateway within the framework that would simplify the use of one-to-many multicast. However, with active replication, malicious states might be replicated as well. Vulnerabilities and breaches that had been found in the primary node would be replicated in the replica as well which would not serve our purpose.

The checkpointing approach combined with an Intrusion Detection System (IDS) may solve the above condition. If on detecting an intrusion or attack through the IDS and failure of the node, the last checkpoint before the intrusion may be restored. However, this approach may slow down legitimate faults that occur through other means.

To detect failure in the nodes, a heartbeat protocol may be used. A heartbeat protocol is already in place between the Poly^2 gateways and a protocol based on the work of (Li et al., 2009) may be implemented with the gateways as the control nodes.

The researcher expects a hybrid active replication-checkpointing approach as a way towards solving our research question. As a part of this, it is expected to make this feature a part of the Poly^2 framework.

3.2. Cost-Benefit Analysis

The cost-benefit analysis of the project was carried out based on the benefits with respect to the alternatives available. We would be looking at active replication and checkpointing as alternatives since they are the ones most popular in usage today. Our costs would be based on the impact on the original design of the Poly^2 framework, the engineering resources which would be involved and the software engineering effort which would be involved. Our benefits would be in terms of decrease in downtime and the effectiveness of the recovery that can be initiated.

With respect to the monetary costs regarding the costs as well as benefits with respect to the projects analyzed, costs are recorded on a per incident basis leading to an average dollar value (Gordon et al, 2005). However, the Poly^2 framework is yet to be implemented commercially giving it a unique perspective. Due to it's current prototype status and lack of public deployment, obtaining precise values for dollar costs would be beyond the scope, hence the analysis would be restricted to non-monetary costs. The analysis restricts itself to the first order effects of the costs as well as the benefits. That is, we consider the effects only on the actors who would be affected through direct loss of business, availability and reputation if the service became unavailable.

CHAPTER 4. COST BENEFIT ANALYSIS

As we have mentioned before, for the purpose of our cost benefit analysis we will be looking at cost benefit analysis of the project through two aspects. First, we will be looking at the alternatives. These include two recovery paradigms, one is based on the active state replication paradigm and the other is based on a checkpointing paradigm. Second, we will be looking at the alternative suggested as a part of this project and comparing it to the other two alternatives. This analysis was carried out in the *media res* form. In this case, i.e. it was carried out in the duration of the project.

4.1. Analysis of the Active State Replication Paradigm

The active state replication paradigm relies on a single primary server and multiple replicas of the server. Under normal functioning, the state of the primary service is replicated across the replicas. Whenever the primary server is subjected to a fault, one of the replicas takes over in a process that is transparent to the user. There are two basic ways this technique can be implemented. One, a proxy or gateway handles communication between the users and service. The proxy keeps track of the availability of the service and distributes the incoming communications to all the nodes but allows only the communication from the active node to go out. When a fault occurs, it redirects traffic from the next active node. The other type in which there is no proxy, all the nodes receive the traffic and use it to update their states. However, it is only the active node that responds. When it fails, the next node takes over. In case of a proxy, there has to be a significant change to the Poly^2 architecture to accommodate another gateway in front of the services. In case of no proxy, it becomes the responsibility of the nodes internally to keep track of the active node. Also, to ensure that all the nodes receive the incoming data, the multicast has to be reliable.

For both forms of the technique, the communication costs will be high since the states are being kept perennially updated. Putting up either a proxy or making the multicast reliable will require a high engineering cost. In case an exploit occurs, it is replicated across the nodes and persists even if the initial active node fails and the new one takes over.

This technique however, offers lower downtime. Since the next node is already active and in the current state of the service, it can take over seamlessly. Also, due to state persistence, when the active node fails, the data is preserved by all the nodes in the replication cluster. The storage required is limited to any write action carried out on the nodes.

Hence, the high costs are associated with communications, engineering and exploitation persistence. Costs are mitigated in terms of loss of data, storage, downtime and loss of data.

4.2. Analysis of the Checkpointing Paradigm

The checkpointing paradigm ensures that service on restoration is always restored from a point at which the service was uncompromised. A checkpoint is maintained on which the image of the service is kept and if required updated. When a fault occurs, the service is restored from the stored checkpoint. The storage costs with regard to this are sufficiently high since a complete image will have to be kept to initiate a recovery. Since the checkpoint is not necessarily a reflection of the current state of the service, there may be a significant amount of data loss when the recovery from checkpoint is initiated. There may be a perceptible difference in time as the checkpointing showing a small but significant downtime.

However, the communication costs would be low since the active node would not need to communicate with any other node. The checkpoint can be hosted as a virtual machine allowing restoration from an image. This decreases the engineering effort required to carry it out. Since, a checkpoint need only be carried out infrequently and then at the point of a fault, the processing overheads involved are low. Since the current state is not maintained and the checkpoint is restored from an uncompromised state, any exploitation on the service carried out in the current state does not persist post-recovery.

Hence, the costs are associated in terms of storage, loss of data and downtime. Costs are mitigated in terms of communication, engineering effort, processing overheads and persistence of exploitations.

4.3. Analysis of the Hybrid Recovery System

The hybrid system creates a two-tier recovery technique including aspects of both the active state replication as well as the checkpointing recovery system. It invokes the active state replication when there are faults that may not have been triggered by a malicious entity. In this case, one of the replicas would take over the function of the primary service, maintaining the state and ensuring that the process is transparent to the user. However, in case of exploits caused by a malicious entity the service would be restored from a previous clean state ensuring that service being provided is uncompromised. In case of this technique, the average downtime would be lower since a checkpoint recovery would only be triggered in case of a malicious activity being triggered by a detection system. Also unlike the active replication system, the state would only be maintained in case of fault that has not been triggered by a malicious entity. This would ensure that the exploitation of vulnerabilities which may have been persistent over the replicas is not so.

However, the costs incurred in this case would be the engineering efforts involved in setting up this technique as well as the cost of setting up an intrusion detection system to determine which faults may have been triggered by a malicious entity. Also involved would be the cost of implementing both the checkpointing recovery technique as well as the active state replication technique including the human effort involved. In spite of the costs involved, the critical services that fall within the purview of the Poly^2 framework would benefit greatly from the hybrid technique. Availability of services often affects events such as loss of business and reputation that is significant for the actors involved.

4.4. Conclusion

From the above analysis, we can observe that there are a number of factors on which the costs can be obtained. These costs and their possible mitigation can guide the reasoning for selecting a particular recovery technique. This, however, does not take into account the costs or benefits with regard to the adoption of the Poly^2 framework. This analysis was carried out with respect to services that are being run under the Poly^2 framework.

With regard to costs of communications, it is seen to be high in case of active replication and low in case of checkpointing. It is high in case of active replication due to the communication required to maintain the state of the nodes. The same cost applies to the hybrid recovery technique as well.

With regard to costs of storage, this cost is significantly high in case of the checkpointing technique due to the requirement to store the image of the service. This cost would apply to the hybrid recovery technique as well since the checkpoint technique is implemented as a part of the hybrid recovery technique.

The active state replication has a high engineering cost as well as a possibly high cost due to the impact on the framework. The hybrid recovery technique also has the same cost associated. However, instead of using a proxy that could potentially increase the impact on the framework itself, the proxy-less alternative is used. This cost is however; relatively low for the checkpointing technique where the service is hosted upon a virtual machine and restored from an uncompromised image upon the occurrence of a fault. The similar cost for the hybrid recovery mechanism is high since it is required to have both the functions of the checkpointing as well as the active replication technique.

The cost occurring due to loss of data is high with the checkpointing technique. This is due to the fact that when it is initiated, the service is restored to a previous state and all the data in the interim is lost. This is mitigated with respect to the hybrid recovery technique. In this case, the checkpointing is initiated only when a malicious fault is thought to have occurred.

The cost associated with downtime is significantly higher in checkpointing. This is mitigated in the hybrid recovery technique. This is again due to the fact that the checkpointing is initiated only in the specific circumstance of a malicious fault. Otherwise, the technique uses active replication to recover from a fault that has a lower cost associated with downtime.

The cost associated with processing overheads is high with the active replication technique since it requires that the state be constantly upgraded across all nodes. This cost is reflected in the hybrid recovery as well.

The persistence of exploitation is present in the active replication technique. However, it is significantly mitigated with the hybrid recovery technique. In case of malicious faults, the technique uses the checkpointing from a previously uncompromised state where the exploitations are no longer allowed to persist.

Table 4.1 Table of Costs

Cost / Technique	Communication	Storage	Engineering	Loss of Data	Downtime	Processing Overheads	Exploitation Persistence
Active Replication	High	Low	High	Low	Low	High	High
Checkpointing	Low	High	Low	High	High	Low	Low
Hybrid Recovery	High	High	High	Medium	Medium	High	Medium

With respect to the benefits, it is seen that the active replication technique has a lower time to recovery in comparison to the checkpointing technique. However for the hybrid recovery technique, it is seen that while time to recovery is low when it is functioning in the active replication form, it may increase when a malicious fault occurs and the checkpointing form is employed.

In case of the state preservation and continuity of the service, the hybrid recovery technique maintains the state in case of normal functioning, i.e. when no faults occur and when there are non-malicious faults. In case of the checkpointing technique, the current state is not preserved and on recovery, current connections and data may be lost.

In case of removal of exploits, when a malicious fault occurs, the hybrid recovery technique initiates the checkpointing that restores the service from an uncompromised image. However, an active replication technique replicates the exploit over all the nodes and when the current active node fails, the node to take over has the same exploit.

Both the checkpointing and active replication respond to all incidents in the same manner. However, the hybrid recovery mechanism differentiates the response based on the input from the intrusion detection system. If a non-malicious fault occurs, the active replication form is triggered. If a malicious fault is detected, the checkpointing technique is initiated. This ensures that while the state is preserved and normal functioning can be carried out in cases of non-malicious faults but in case of malicious ones, the exploits are not allowed to remain persistent and all connections including potentially malicious ones are abandoned.

Table 4.2 Table of Benefits

Benefit / Technique	Time to Recovery	Preservation of State	Removal of potential exploits	Response to incident
Active Replication	Low	High	Low	Low
Checkpointing	High	Low	High	Low
Hybrid Recovery	Medium	High	High	High

We see that the hybrid recovery technique shares the costs with regard to both the active replication as well as the checkpointing technique. However, it also mitigates the costs with respect to both leading to reduced loss of data and downtime (in comparison to checkpointing) and reduced persistence of exploits (in comparison to active replication). It also is able to provide a differentiated response in case of both malicious and non-malicious faults and thus, providing an alternative to both active replication and checkpointing.

CHAPTER 5. THE ARCHITECTURE

To implement the technique, a two-tier architecture was implemented. To recover from non-malicious faults, it is considered valuable if the state is maintained. This means that the recovery would need to be such that loss in capacity is transparent to the service user. However for malicious faults, such a capacity may not be wholly in favour of the service. A state of being actively replicated may not be advisable in the case of a malicious fault. In such a case, it might be required that the connections be abandoned and restored to a previously uncompromised state. Hence, it would be important that the potentially malicious connection is abandoned and the service restored. This is carried out by restoring the service from a backup in the form of an image. There is, however, another issue of detecting when a malicious fault occurs. For this purpose, we use an intrusion detection system (IDS) to generate the alerts whenever a potentially malicious activity occurs.

To maintain the state in case of non-malicious faults, replicas of the service are kept whose states are kept consistent with that of the primary service. In the event of the failure of the primary service, one of the replicas may be able to take over in a process that is transparent to the user. Due to the maintenance of the state, current connections are not lost and neither is the data.

Next, the three main components of the architecture would be explored. The checkpointing system that provides recovery from malicious would be first explored. Then, the active state replication would be looked at. After that, the detection system that bridges these two recovery systems would be looked at.

5.1. Checkpoint Recovery

The recovery system suggested is similar to the one proposed by Laadan et al (2005). However, this system has significant changes to accommodate the requirements of the project including the limitations and the scope already stated. To carry out the checkpointing, another layer introduced between the host operating system and the service. The service itself is hosted on a virtual machine that is in turn hosted on the server. The servers used are of two kinds – Sun SunFire V60X and Dell PowerEdge 1850.

The SunFire V60X has the host operating system as FreeBSD 8.0 running under the following hardware specifications:

- Processor (1): Intel Xeon (2.8 Ghz)
- RAM: 1024 MB
- Cache: 512 KB L2 cache
- Networking: Dual 10/100/1000 NIC

The Dell PowerEdge 1850 has the host operating system as FreeBSD 8.0 running under the following hardware specifications:

- Processor (1): Intel Xeon (2.8 Ghz)

- RAM: 1024 MB
- Cache: 1024 KB
- Networking: Dual 10/100/1000 NIC

The service itself is run on virtual machines on a Sun VirtualBox platform administered across the network by the Poly^2 security server. The running of the checkpointing technique is shown below.

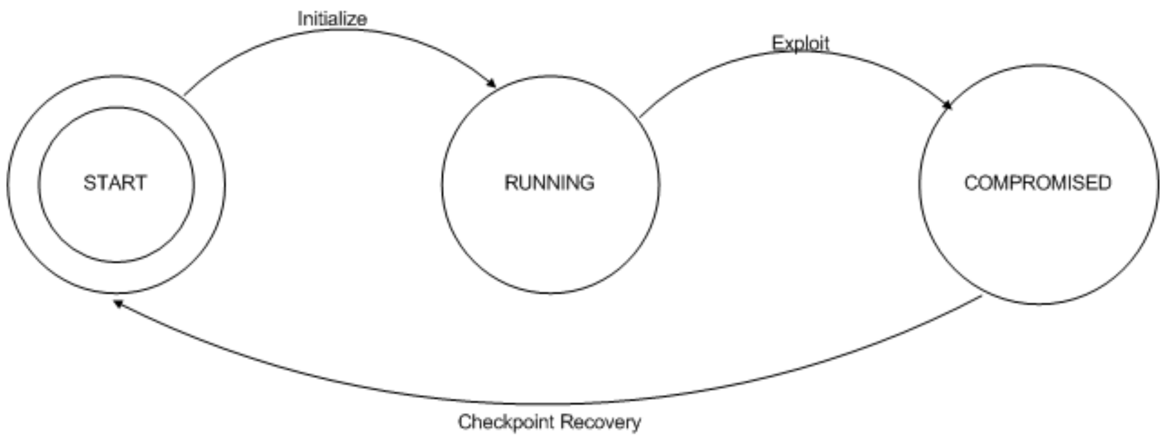
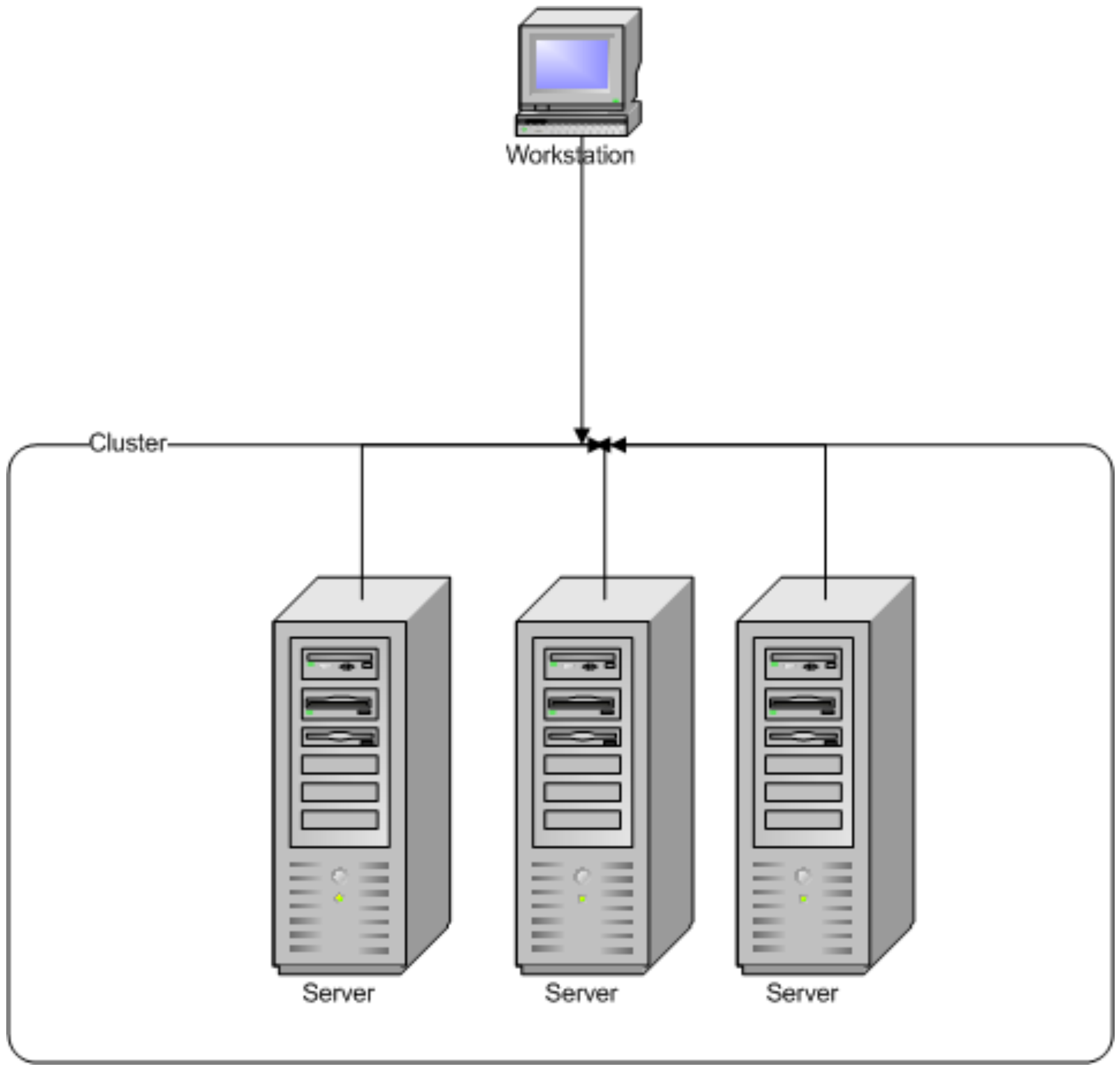


Figure 5.1 Checkpointing form

When the service is initialized, a image of it is created and kept. Assuming that initially image was not compromised, this image is kept as an uncompromised state. When the compromise of a service is detected, the current state of the service is abandoned and the service is restored from the uncompromised image. However, as a result of this all current connections are abandoned and any data that was in the current state is also. In case of the chat server implementation, this refers primarily to the chat history and the connections to the clients. When the service is compromised and a recovery is carried out from a checkpoint, the connections as well as the chat history are lost.

5.2. Active Replication

The replication for maintaining the state of the application is carried out at an application level. For the purpose of the project, the application to be used for the purpose is chat server that has been replicated across two servers. This would be similar to the approach taken by Wang et al (2001). The application itself built upon the foundation provided by the JGroups API. JGroups is a toolkit for providing reliable multicast communication. This property is used to provide maintenance of state between the primary service and it's replicas. Interactions take place between the client and the cluster hosting the services. The cluster is transparent to the user. Upon failure of the primary server, a replica takes over it's functions in a manner that the previous sessions are carried over. Such a function while desirable in the case of a non-malicious failure, however, in case of the involvement of a malicious entity, the recovery would revert to the checkpoint system enumerated formerly. The active replication form is shown below.



The user interacts with the service in a manner that the active replication form is transparent to it. Whenever there is a communication from the user to the active node, all the nodes intercept it but the active node only sends the response. This ensures that the state (in this case, the chat history) is current across all the nodes. Whenever the current active node fails, the operation can be taken over by the next node.

5.3. Detection

There has to be a differentiation as to when of the above two techniques has to be used. While the active replication technique maintains the state of the service but the technique requires a higher overhead and also leads to persistence of any exploits that take place. The checkpoint technique ensures that the system is restored from a clean state, however, with significant loss of data and higher recovery time.

To differentiate which of the two techniques should be employed according to the event, an intrusion detection system was employed. On detection of an event by the system, the checkpoint recovery is triggered.

In case of this project, an intrusion detection system based on SNORT was employed. It was hosted upon a hardened system having a FreeBSD 8.0 operating system. The hardware it was hosted upon was a Sun SunFire V60x.

The intrusion detection system was placed in front of the primary gateway to and was made transparent to the services as well users accessing the services.

5.4. Chapter Conclusion

The architecture that has been enumerated is overlayed upon the Poly^2 framework. It does not affect the original design of the Poly^2 framework but extends it. All services are still isolated from each other not only in their own individual servers but in their individual clusters. Attacks taking place on a specific service are not only isolated to the particular service but the service itself is not completely lost as well with recovery mechanisms being in place. The complete framework is seen below.

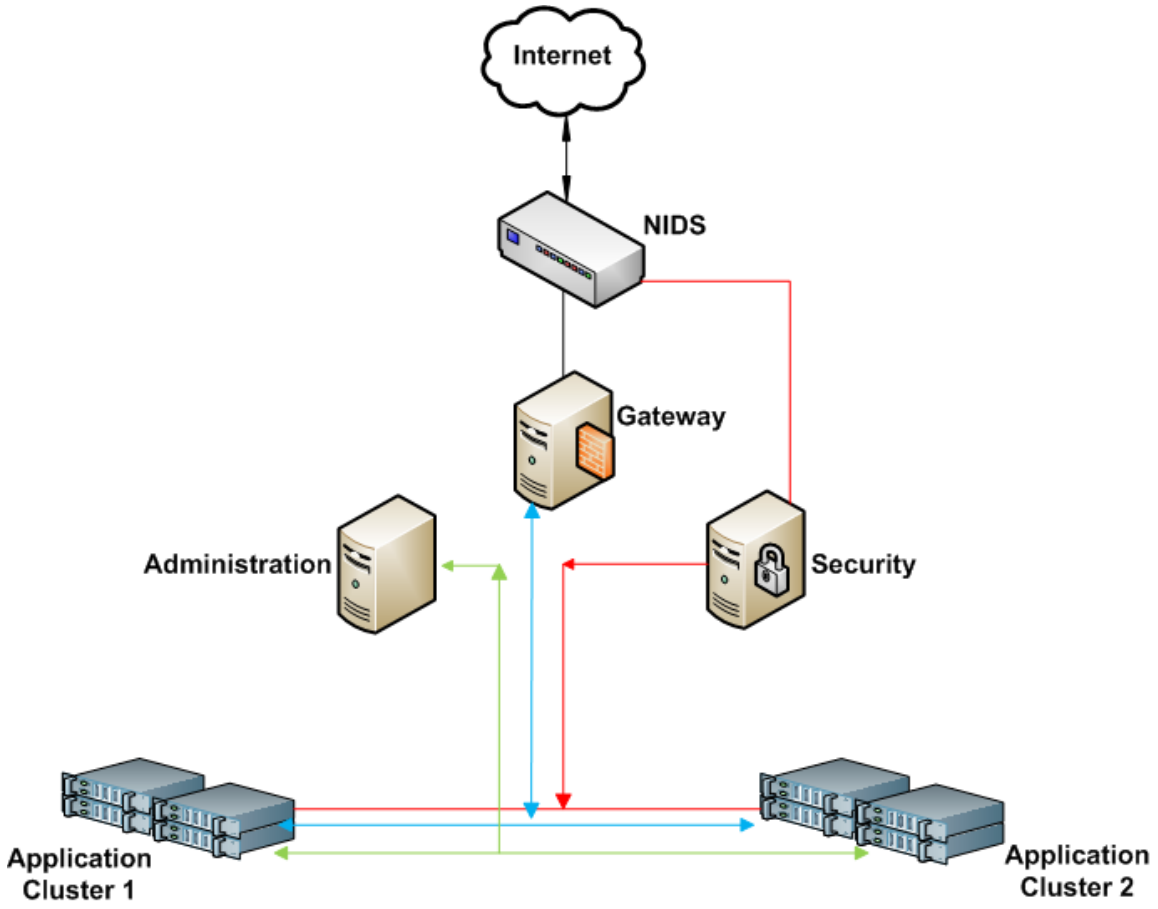


Figure 5.3 Poly^2 with Availability

The application clusters shown have the recovery mechanisms for the particular service. Each of the services is contained within a single cluster instead of a single server. The nodes in each cluster provide for the active replication while each node has the capacity to carry out a checkpoint recovery on direction from the security server. The security server also monitors the intrusion detection system and upon receiving a alert, it instructs the nodes of attacked service to undergo checkpointing.

CHAPTER 6. CONCLUSION AND FUTURE WORK

The recovery mechanisms that have been explored in this project add a novel extension to the Poly^2 framework. The original framework isolated each of the services into their own ‘cages’. A cage was their particular hardware and operating system allowing for their isolation to the outside attacks. The recovery mechanisms acted as an extension to this concept. Each of the service cages is extended to be a cluster to provide recovery to the service.

The recovery mechanism suggested in this work is novel in the sense that it takes the advantages of two well-known mechanisms and applies them in a manner that mitigates their disadvantages. The active state replication ensures that in case of non-malicious faults, the user experiences only a minimal amount of disturbance. However, the checkpointing recovery system ensures that in case of malicious faults, the system recovers from it albeit with an increased amount of downtime and loss of data.

This system is meant to be used under two conditions. The service has to be placed under the Poly^2 framework. The service also has to be such that it is expected to be attacked occasionally and not with a very high frequency. This was shown by the cost-benefit analysis where it was seen that a high occurrence of the checkpointing form, which is employed in case of malicious faults, will lead to a higher cost in terms of downtime and loss of data.

However, there are some issues that can be addressed. Currently, the recovery in the checkpoint mechanism takes place from a former, static version of the service. Whenever, the service is restored from an uncompromised image, it is still vulnerable to the same vulnerability that previously exploited the service. This can be improved by having a mechanism that on regular intervals updates the service image so that on recovery at some point, the service restored is a fixed version of the image.

The intrusion detection system is a rule-based system that detects breaches based upon the ruleset upon which the system functions. However, if the breach is not detected by the intrusion detection system, it may lead to a fault that the system may believe to be non-malicious which will eventually lead to an incorrect part of the mechanism being initiated. A detection system that is more robust may improve the functioning of the overall system.

The active replication part of the system is currently taking place at an application level. Improving this in a manner such that this takes place at a level independent of the application allowing this system to be generic for any application that would need to be placed within the framework.

The recovery system though hybrid in nature can be improved by changing the very essence of it. The current system requires that the functions be differentiated on the basis of input received from the detection system. A unified recovery paradigm that can provide comprehensive recovery within the paradigm of the Poly^2 framework would be an interesting improvement to the project.

The Poly^2 framework is meant to protect an organization’s mission-critical services. However, setting up such a framework for an organization can be an expensive undertaking. It would be an interesting exploration to look at an approach where the framework is offered as a service where mission-critical services can be placed in their individual Poly^2 ‘cages’.

BIBLIOGRAPHY

BIBLIOGRAPHY

Anderson, T., & Lee, P. (1982). Fault tolerance terminology proposals. In *Proceeding of the 12th International Symposium on Fault Tolerant Computing*.

Ayari, N., Barbaron, D., & Lefevre, L. (2008). On Improving the Reliability of Internet Services through Active Replication. In *Parallel and Distributed Computing, Applications and Technologies, 2008. PDCAT 2008. Ninth International Conference on* (pp. 259-262). Presented at the Parallel and Distributed Computing, Applications and Technologies, 2008. PDCAT 2008. Ninth International Conference on. doi:10.1109/PDCAT.2008.82

Ayari, N., Barbaron, D., Lefevre, L., & Primet, P. (2008). Fault tolerance for highly available internet services: concepts, approaches, and issues. *Communications Surveys & Tutorials, IEEE*, *10*(2), 34-46. doi:10.1109/COMST.2008.4564478

Bhide, A. (1992). Experiences with two high availability designs [replication techniques]. In *Management of Replicated Data, 1992., Second Workshop on the* (pp. 51-54). Presented at the Management of Replicated Data, 1992., Second Workshop on the. doi:10.1109/MRD.1992.242618

Bryant, E., Early, J., Gopalakrishna, R., Roth, G., Spafford, E., Watson, K., Williams, P., et al. (2003). Poly^2 paradigm: A secure Network Service Architecture. Presented at the Annual Computer Security Applications Conference.

Cachin, C., & Poritz, J. (2002). Secure INtrusion-Tolerant Replication on the Internet. In *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on* (pp. 167-176). Presented at the Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on. doi:10.1109/DSN.2002.1028897

Chee, J. (2008). *Host Intrusion Prevention Systems and Beyond*. SANS Institute.

Deering, S. (1989, August). RFC 1112: Host Extensions for IP Multicasting. Network Working Group.

Deplanche, A., Theaudiere, P., & Trinquet, Y. (1999). Implementing a semi-active replication strategy in CHORUS/ClassiX, a distributed real-time executive. In *Reliable Distributed Systems, 1999. Proceedings of the 18th IEEE Symposium on* (pp. 90-101). Presented at the Reliable Distributed Systems, 1999. Proceedings of the 18th IEEE Symposium on. doi:10.1109/RELDIS.1999.805086

Engelmann, C., Scott, S. L., Leangsuksun, C., & He, X. (2007). On Programming Models for Service-Level High Availability. In *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on* (pp. 999-1008). Presented at the Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on. doi:10.1109/ARES.2007.109

Fei-fei Li, Xiang-zhan Yu, & Gang Wu. (2009). Design and Implementation of High Availability Distributed System Based on Multi-level Heartbeat Protocol. In *Control, Automation and Systems Engineering, 2009. CASE 2009. IITA International Conference on* (pp. 83-87). Presented at the Control, Automation and Systems Engineering, 2009. CASE 2009. IITA International Conference on. doi:10.1109/CASE.2009.115

Garbin, D., & Knepley, J. (2009). Design and analysis of high availability networks. In *Technologies for Homeland Security, 2009. HST '09. IEEE Conference on* (pp. 1-6). Presented at the Technologies for Homeland Security, 2009. HST '09. IEEE Conference on. doi:10.1109/THS.2009.5168007

Geraint Price, C Geraint Price, & Geraint Price. (1999). The Interaction Between Fault Tolerance and Security. Retrieved from HTTP://CITESEERX.IST.PSU.EDU/VIEWDOC/SUMMARY?DOI=10.1.1.62.8328

Gordon, L. A., Loeb, M. P., Lucyshyn, W., & Richardson, R. (2005). The 2005 CSI/FBI computer crime and security survey. *Computer Security Journal*, *21*(3), 1.

Gouda, M., & McGuire, T. (1998). Accelerated heartbeat protocols. In *Distributed Computing Systems, 1998. Proceedings. 18th International Conference on* (pp. 202-209). Presented at the Distributed Computing Systems, 1998. Proceedings. 18th International Conference on. doi:10.1109/ICDCS.1998.679503

Hamada, M. (1995). Using statistically designed experiments to improve reliability and to achieve robust reliability. *Reliability, IEEE Transactions on*, *44*(2), 206-215. doi:10.1109/24.387372

Information Processing Systems - Interconnection Reference Model - Part 2: Security Architecture. (1989). . International Standards Organization.

Information Technology - Code of practice for information security management. (2000). . International Standards Organization.

Information Technology - Security Techniques - Guidelines for the management of IT Security. (2001). . International Standards Organization.

Kalbarczyk, Z., Iyer, R., Bagchi, S., & Whisnant, K. (1999). Chameleon: a software infrastructure for adaptive fault tolerance. *Parallel and Distributed Systems, IEEE Transactions on*, *10*(6), 560-579.

Kyamakya, K., Jobman, K., & Meincke, M. (2000). Security and survivability of distributed systems: an overview. In *MILCOM 2000. 21st Century Military Communications Conference Proceedings* (Vol. 1, pp. 449-454 vol.1). Presented at the MILCOM 2000. 21st Century Military Communications Conference Proceedings. doi:10.1109/MILCOM.2000.904993

Laadan, O., Phung, D., & Nieh, J. (2005). Transparent Checkpoint-Restart of Distributed Applications on Commodity Clusters. In *Cluster Computing, 2005. IEEE International* (pp. 1-13). Presented at the Cluster Computing, 2005. IEEE International. doi:10.1109/CLUSTR.2005.347039

Laprie, J. (1992). *Dependability: Basic Concepts and Terminology*. Springer-Verlag.

Lundell, J. (2001). A fault-tolerant approach to network security. In *Network Computing and Applications, 2001. NCA 2001. IEEE International Symposium on* (p. 227). Presented at the Network Computing and Applications, 2001. NCA 2001. IEEE International Symposium on. doi:10.1109/NCA.2001.962536

Malkhi, D., & Reiter, M. (1998). Secure and scalable replication in Phalanx. In *Reliable Distributed Systems, 1998. Proceedings. Seventeenth IEEE Symposium on* (pp. 51-58). Presented at the Reliable Distributed Systems, 1998. Proceedings. Seventeenth IEEE Symposium on. doi:10.1109/RELDIS.1998.740474

Meadows, C. (1995). Applying the dependability paradigm to computer security. In *New Security Paradigms Workshop, 1995. Proceedings* (pp. 75-79). Presented at the New Security Paradigms Workshop, 1995. Proceedings. doi:10.1109/NSPW.1995.492346

Meadows, C. (1994). The Need for a Failure Model for Security. Article, . Retrieved October 15, 2009, from HTTP://EPRINTS.KFUPM.EDU.SA/70629/

Neumann, P. (2000). *Practical Architectures for survivable systems and networks* (Technical Report Phase Two No. Project 1688). Menlo Park, California: SRI International.

Price, G. (2001). Broadening the Scope of Fault Tolerance within Secure Services. In *Security Protocols* (pp. 155-164). Retrieved from HTTP://DX.DOI.ORG/10.1007/3-540-44810-1_20

Reiter, M. K. (1995). The Rampart Toolkit for Building High-Integrity Services. In *Selected Papers from the International Workshop on Theory and Practice in Distributed Systems* (pp. 99-110). Springer-Verlag. Retrieved from HTTP://PORTAL.ACM.ORG/CITATION.CFM?ID=723763

Ross, S. (1997). *Introduction to probability models* (6th ed.). Academic Press.

RossebeØ, J. E. Y., Lund, M. S., Husa, K. E., & Refsdal, A. (2006). A Conceptual Model for Service Availability. In *Quality of Protection* (pp. 107-118). Retrieved from HTTP://DX.DOI.ORG/10.1007/978-0-387-36584-8_9

Rosseboe, J., & Braek, R. (2006). Towards a framework of authentication and authorization patterns for ensuring availability in service composition. In *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on* (p. 10 pp.). Presented at the Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on. doi:10.1109/ARES.2006.135

Saltzer, J., & Schroeder, M. (1975). The protection of information in computer systems. In *Proceeding of the IEEE* (Vol. 63, pp. 1278-1308).

Schmidt, D., & O'Ryan, C. (2000). *Leaders/Followers, A design pattern for efficient multi-threaded I/O demultiplexing and dispatching*. Siemens.

Schneider, F., & Zhou, L. (2005). Implementing trustworthy services using replicated state machines. *Security & Privacy, IEEE*, *3*(5), 34-43. doi:10.1109/MSP.2005.125

Telecom Glossary. (2007). . Alliance for Telecommunications Industry Solutions.

Wang, L., Zhou, W., & Jia, W. (2001). The design and implementation of an active replication scheme for distributing services in a cluster of workstations. *Journal of Systems and Software*.
Xinyu Chen, & Lyu, M. (2005). Reliability analysis for various communication schemes in wireless CORBA. *Reliability, IEEE Transactions on*, *54*(2), 232-242. doi:10.1109/TR.2005.847268

Yen, I., Ahmed, I., Jagannath, R., & Kundu, S. (1998). Implementation of a customizable fault tolerance framework. In *Object-Oriented Real-time Distributed Computing, 1998. (ISORC 98) Proceedings. 1998 First International Symposium on* (pp. 230-239). Presented at the Object-Oriented Real-time Distributed Computing, 1998. (ISORC 98) Proceedings. 1998 First International Symposium on. doi:10.1109/ISORC.1998.666793

Zhanjie Wang, & Xiao Li. (2008). A New Real-Time Heartbeat Failure Detector. In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on* (pp. 1-3). Presented at the Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on. doi:10.1109/WICOM.2008.1233

Boardman, A., Greenberg, D., Vining, A., & Weimer, D. (2005). *Cost Benefit Analysis: Concepts and Practice* (3rd ed.). Prentice Hall.

Gordon, L., & Loeb, M. (2005). *Managing Cybersecurity Resources: A Cost-Benefit Analysis* (1st ed.). McGraw-Hill.

Sassone, P. G. (1988). Cost benefit analysis of information systems: A survey of methodologies. *ACM SIGOIS Bulletin*, *9*(2-3), 126–133.

Wei, H., Frinke, D., Carter, O., & Ritter, C. (2001). Cost Benefit Analysis of Network Intrusion Dertection System. *CSI 28th Annual Computer Security Conference October 29-31, 2001 Washington, DC*.