Survivability Architecture of a Mission Critical System: The DPASA[1] Example

Jennifer Chong, Partha Pal, Michael Atigetchi, Paul Rubel, Franklin Webber
BBN Technologies, Cambridge, MA
{jchong, ppal, matiget, prubel, fwebber}@bbn.com

Abstract

Many techniques and mechanisms exist today, some COTS, others less mature research products that can be used to deflect, detect, or even recover from specific types of cyber attacks. None of them individually is sufficient to provide an all around defense for a mission critical distributed system. A mission critical system must operate despite sustained attacks throughout the mission cycle, which in the case of military systems, can range from hours to days. A comprehensive survivability architecture, where individual security tools and defense mechanisms are used as building blocks, is required to achieve this level of survivability. We have recently designed a survivability architecture, which combined elements of protection, detection, and adaptive reaction; and applied it to a DoD information system. The resulting defense-enabled system was first evaluated internally, and then deployed for external Red Team exercise. In this paper we describe the survivability architecture of the system, and explain the rationale that motivated the design.

1 Introduction

Previous investments in cyber security, information assurance, and intrusion tolerance research resulted in several solutions for particular cyber threats. For instance, firewalls effectively block unwanted traffic, digital signatures detect modification of data in transit, and redundancy and Byzantine protocols tolerate corrupt or compromised application components. However, these technologies are point solutions and are not adequate by themselves to build a system that can withstand a wide range of threats.

Experience shows that protecting a system against all possible cyber attacks is not currently feasible when the system is composed of COTS components with unknown or unverified security properties. The increasing demands for systems to interconnect and interoperate with each other exacerbate the situation. It must be assumed that some attacks will be successful, causing undesirable effects such as loss of availability, integrity, and confidentiality (LOA, LOI, and LOC respectively) in parts of the system. Detection complements protection, but detection alone has known problems such as false positives, false negatives, late detection, missing sensors, difficulties in interpreting alerts, and not having a course of action to fix the problem.

Knowing that attacks will occur and detection could be incorrect or happen too late, we have been investigating how to allow mission critical systems to be cyber-attack tolerant so that they can recover from attack induced failures or to continue to operate in a degraded mode.

Building a system that is resilient, despite a wide range of sustained attacks, requires integration of tools and mechanisms that provide protection and detection, as well as adaptive tolerance. This integration of solutions from each category provides complimentary, defense-in-depth coverage protecting all layers and aspects of the system. We define this integration as a 'survivability architecture', which denotes the well-defined organization, placement, and interconnection of the multiple layers of these security solutions and how they are integrated with the original components of the undefended system.

Continuing with our history of research in dynamic security, adaptive intrusion tolerance, and survivability [1] [2], we recently designed a survivability architecture for a DoD information system, and used the design to build the survivable version of the system in the DPASA project under the DARPA OASIS Dem/Val program.

In this paper, we first present the principles that motivated the design of the survivability architecture. Then we describe the survivable system, explaining how the architecture integrated with and transformed the undefended system. Next, we describe how the system counters attacks. After a short discussion of related work, the paper concludes with a brief summary of the technical contributions of this paper, and a few areas for future research.

[1] DPASA stands for Designing Protection and Adaptation into a Survivability Architecture.

This work has been supported by DARPA under contract number F30602-02-C-0134.

2 Motivating Principles

The high level strategy underlying our approach to designing a survivable system is to strategically combine elements of *protection, detection, and adaptive reaction* in the architecture of the system, such that

- *Protection* is the first line of defense, employing techniques to fortify key aspects of the system to deny the attacker easy or *undetected* access to targets
- *Detection* provides insight into the status of the system and allows the system to detect attacks
- *Adaptive reaction* enables the system to cope with (undesirable) changes caused by attackers exploiting gaps in the protection or circumventing it, to support *recovery* and *graceful degradation*

The need to combine protection, detection, and adaptive reaction is based on the understanding that protection cannot be perfect, some attacks will succeed (at least partially), and some of the attacks will not be detected in time.

When combined with the collective experience of our team members [1][2][3][10], this high level strategy led to a number of design principles that influenced the survivability architecture. These are explained in the remainder of this section.

Multiple Layers of Protection: To maximize the level of protection, a high barrier of entry against break-in to the system needed to be built. A single layer of protection is not sufficient to provide a high barrier, thus multiple layers of protection are needed. Techniques that are grounded in the hardware are suited for providing multiple layers of protection. Autonomic Distributed Firewall (ADF) [3] Network Interface Card (NIC) based distributed firewalls that filter packets, or smart card based mechanisms that control physical access to sensitive data such as private keys are good examples of these techniques. Combining hardware, cryptography, operating system protection domains, and application level session and quorum checking provide multiple layers of protection. This will deter script kiddies, and will also force serious attackers to spend more time and resources to gain entry into the system.

Redundancy and Static Diversity: A secure system cannot have single points of failure in either its critical functionality or its key defensive infrastructure. Redundancy is a fundamental technique used to eliminate single points of failure, but homogeneous redundancy incurs the risk of correlated/common mode vulnerabilities. These vulnerabilities can lead to rapid deterioration of the system, with little recourse. Diversity can be used to manage this risk.

Physical Constraints in the Architecture to Impose Modular Isolation and Containment: A system should make use of architectural constraints and design constructs that will slow the attacker's progress through the system. Even if he manages to gain access to some parts of the system (i.e., overcomes the multiple layers of protective barriers to that part of the system), his privileges in the system and his ability to see and reach other parts of the system should remain severely constrained. Techniques such as Demilitarized Zones (DMZs) can be used to force the attackers to compromise several perimeters before they get to a core asset. Network architecture connecting key resources inside the organizational firewall can be made more compartmental, allowing physical connection only where needed.

Detection and Correlation: When COTS components with unverified security properties are deployed, it is not always possible to detect intrusions in a timely and accurate manner. A system must strive to increase the attacker's risk of being detected. Forcing an attacker to spend more time executing his attack, in conjunction with instrumenting the architecture to sense anomalies and policy violations can significantly raise the attacker's risk. Correlating policy violation reports yield higher accuracy and lower false alarm rates. For example, the absence of heartbeats from component X does not always mean that X has failed or is corrupt. Correlating that absence with the file system corruption of X or a violation of an application-level protocol by X gives much higher confidence that an attack has corrupted X.

Adaptive Response: Adaptive reaction, entailing dynamic response (autonomic and human assisted) to security events is an essential enabler for reaching high survivability. Improved detection and awareness can lead to better reaction—which means better response to observed events and anomalies. Adaptive responses can be used to mask or recover from the effects of attacks or to degrade gracefully, retaining some service for the longest possible time. Examples of adaptive response are managing redundancy and protection mechanisms, adding run time diversity by changing system configuration or behavior, dynamically isolating and containing attack effects, and recovering from attack effects by replacing or restarting failed or compromised components.

Design Based on Weak Assumptions: A system that makes strong assumptions about the environment is more vulnerable [4]. For example, if a system's correct behavior depends on the assumption that the communication environment **always** delivers a request for service **exactly once**, then the attacker has a number of ways to put the system in a bad state: he can block the request or send multiple copies of the request by manipulating the communication environment, without having to attack the service requester or service provider components of the system. On the other hand, a system that makes weaker as-

sumptions about the communication environment will be designed to handle a missing request, multiple requests, a missing response, multiple responses, a corrupted response, as well as the desired response. Such a system minimizes the attacker's opportunity to harm the system by simple manipulation of the operating environment.

In the next section we explain how these principles were applied in the creation of the survivability architecture. In addition to these principles, it is important to note that the survivability architecture has incorporated a number of tunable parameters (for instance, amount of redundancy, amount of diversity, number of defensive layers). These tunable parameters allow for cost-benefit trade-offs, making the survivability architecture applicable to contexts that have different survivability and cost requirements from the DoD information system described.

3 Survivability Architecture: The DPASA Example

In the DPASA project we developed the survivability architecture for a Joint Battlespace Infosphere (JBI) exemplar. The JBI concept is being developed at the Air Force Research Laboratory. More details about the JBI concept and implementation can be found on the JBI home page [5]. The JBI's objective is to facilitate quick integration of disparate Air Force applications to support specific missions. JBI uses publish/subscribe communication for flexible integration of distributed applications.

In this section we introduce the JBI exemplar, the mission it supports and its survivability requirements. Then we present a summary of the key aspects of the survivability architecture we designed. Finally, we describe the defense-enabled or survivable version of the JBI exemplar.

3.1 The Undefended JBI Exemplar and Survivability Requirements

The exemplar JBI shown in Figure 1[2] integrates applications for selecting proper targets, monitoring environmental conditions, and creating air-tasking orders (ATOs). A successful mission would involve making the go- no-go decision on an ATO that may have, among other targets, weapons of mass destruction (WMD) sites. The factors that could influence the go or no go decision include whether the ATO had any WMD sites among its targets, the predicted weather condition in the targeted area, the presence of friendly forces nearby, the possibility of other air traffic (such as logistic support) in the theater, and ultimately the possibility of collateral damage (for instance, the chemical plume from the hit WMD site spreading into civilian areas) resulting from executing the ATO.

The undefended system consists of clients connected to a JBI core (platform). The JBI core is shown as a network cloud representing a public IP networking infrastructure like the SIPRNet[3]. The clients interact with each other exchanging Information Objects (IOs) using publish, subscribe and query (PSQ) operations submitted to the JBI core. Besides providing PSQ, the JBI core allows information flows to be managed and access by clients to be controlled.

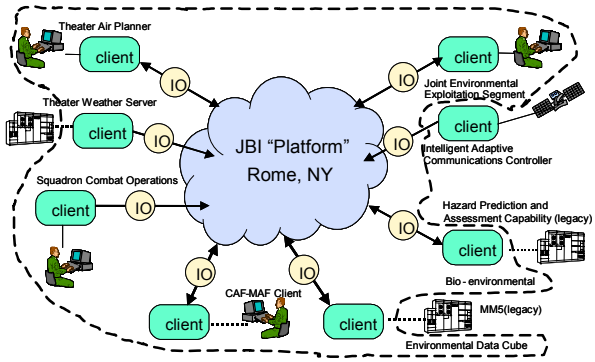

Figure 1 Exemplar JBI - undefended

The defense-enabled JBI is required to survive at least an order of magnitude longer than previous defended systems such as APOD [6] [7]. Additional survivability requirements are: provide 100% of JBI critical functionality under sustained attack by a "Class A" Red Team, detect 95% of attacks in the given time, and ultimately survive 12 hours to finish the JBI mission.

3.2 Overview of DPASA Survivability Architecture

Before we describe the defense-enabled JBI, we give an overview of the key concepts and elements of DPASA architecture.

3.2.1 Zones and Quadrants (Quads)

The DPASA architecture extends the notion of the DMZ using three zones. It replicates the service provider part of the JBI (i.e., the part offering the key services, such as the PSQ service), commonly known as the Core, and protects it by organizing the replicas in quads. As shown in Figure 2, there are four quads and each quad has three zones: crumple, operations, and executive. The crumple

[2] The dashed line delineates the system boundary. Only the elements that are in the system boundary were selected to be defense-enabled.

3 SIPRNet stands for Secret Internet Protocol Router Network. It is the DoD's classified version of the civilian Internet.

zone, which acts as the region of first impact, buffers the core assets in the operations zone. The executive zone sits behind the operations zone and hosts the functions that manage the overall security and survivability.

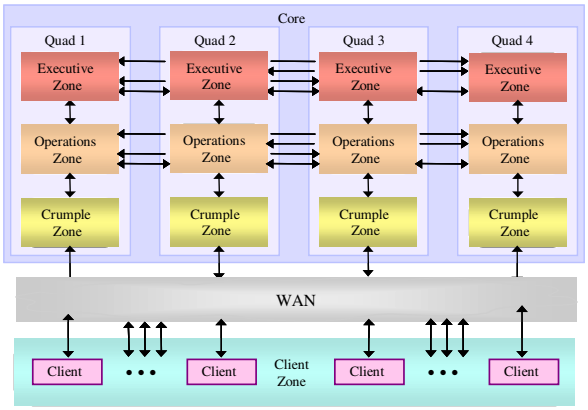

Figure 2 Quads and Zones

The crumple zone consists of the Access Proxy (AP) host. The AP host acts as the entry point for all traffic between the Core from the clients. For each process running on Core hosts that the clients need to interact with, there is a corresponding proxy running on the AP host that performs rate limiting, size-checking, application level checks without violating the end-to-end authentication, and encryption before forwarding traffic to the Core.

The operations zone contains five hosts: the Network Intrusion Detection System (NIDS), the PSQ host, the Policy Server (PS) host, the Correlator (CORR) host, and the Downstream Controller (DC) host. The NIDSes are deployed in each quad. Each NIDS sniffs ingress and egress traffic on the LAN and sends alerts to the CORR. The PSQ host runs the PSQ server. Guardians augment the PSQ server to check the size, the frequency, and the rate of the requests exchanged and report anomalies to the System Manager (SM). The PS host runs processes that manage the ADF NICs in the system, and interfaces with the SM for mounting adaptive responses that make use of the ADF NICs. The CORR host runs processes that receive alerts, generate advice about a host, and correlate the alerts. The correlated alerts are ranked, aggregated, and displayed on a visual interface running on the SM host. The DC host runs the DC component, which processes the heartbeat messages, and acts as an operations zone proxy for the SM so that the SM is not directly exposed to the crumple zone.

The executive zone contains the SM host. The SM host runs the System Manager component that manages the Core components running in the quad, and presents a user interface for the security operators.

On most hosts, there is a set of Local Controllers (LCs) running to monitor the host and to carry out proactive and reactive actions against suspicious activities on the host.

Communication among elements within a zone and across zones is strictly controlled to limit attack progression from compromised parts. Furthermore, quads can be isolated, stopped and restarted. Communication between clients and the processes running on different core zones uses different application level protocols

3.2.2 Protection Domains

In order to implement a barrier to entry and to impede attack propagation, we use protection domains. DPASA uses three protection domains: network, system, and process.

The **Network Protection Domain** protects network traffic, restricts network attack points, and prevents misuse of the network. The **System Protection Domain** protects the host's integrity by preventing modification to critical system files and protects application integrity by preventing modification of critical application resources. It enforces least-privilege at the host level and is deployed on all hosts. Each host process is granted privileges based on the requirements of the system and any attempt to carry out an action (such as accessing the file-system or contacting the network) that is not in the granted privilege is denied and logged. The **Process Domain** enforces additional constraints such as restricting the system calls an application can make.

3.2.3 Protocols

The components described in section 3.2.1 use protocols to communicate with other pertinent components. The major protocols in DPASA are the registration protocol, alert protocols, PSQ protocol, heartbeat protocol, and SM protocols.

A client uses the **registration protocol** when it is ready to join the JBI. To do so, the SM authenticates itself to the client and the client authenticates itself to the SM. The clients and the Core use **alert protocols** to alert the Core of possible issues. The **PSQ protocol** implements publish, subscribe, and query in a fault-tolerant way. When all four quads are participating in the protocol, clients see correct behavior from the Core even if any one of the PSQ servers is corrupt and behaving in an arbitrarily malicious way. To detect the failure of the DPASA survivability components, the **heartbeat protocol** is used. Heartbeat messages are sent to the Core where the SMs use the messages to display the status of the system. The SM uses the **SM protocols** to control ADF NICs, to interact with the PSQ servers to perform quorum group management and other published data management, to take an action on an accused host, or to gather status of components.

3.3 The Defense-Enabled System: The JBI Exemplar Instantiated in the Survivability Architecture

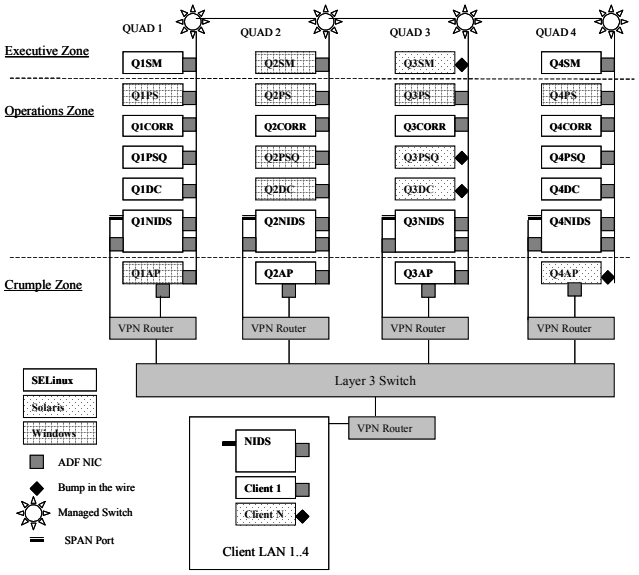

Figure 3 Defense-enabled JBI

Figure 3 shows the defense-enabled JBI. First note that rather than one JBI core LAN, it has four Core LANs (quads). The Core uses redundant service providers. These services perform the same function but are hosted on redundant hardware using three different operating systems (Security Enhanced Linux (SELinux), Windows, and Solaris).

The crumple zone and the executive zone each have one host (denoted by Q*X*AP and Q*X*SM respectively in the figure) in every quad. The operations zone has five hosts (the Q*X*NIDS, the Q*X*DC, the Q*X*PSQ, the Q*X*CORR, and the Q*X*PS) in each quad. The clients are organized into four LANS and the configuration of the client LAN is not changed except for the addition of a NIDS in each LAN.

The network topology of the defense-enabled JBI provides defense-in-depth using multiple layers of protection. There is a VPN router in front of each LAN, establishing a VPN trunk from each client LAN to each of the Core LANs. All hosts are equipped with ADF NICs, as shown by the small solid rectangles. Hosts in the crumple zone have two ADF NICs since each crumple zone host has two network interfaces: one client facing and the other Core facing. A device driver for the ADF card was not available for the Solaris operating system, therefore the Solaris hosts had an ADF equipped Linux host acting as a bump in the wire (indicated by the small diamond attached to Solaris hosts). Each core LAN has a managed switch that controls the communication between quads as well as inter-zone communication within its own quad. Each managed switch is further controlled via a Quadrant Isolation Switch (QIS) by the four SMs (System Managers). The QIS allows security operators to isolate an entire quad if necessary by cutting off the power to the managed switch. A quorum of three SMs is required to isolate a quad. The public IP network is simulated by a layer 3 switch and instead of implementing each client and core LAN as a physical LAN, they are implemented as VLANs off the layer 3 switch.

Host protection domain policies on hosts tightly define how processes can be started, and limit the capabilities a process should have. In addition to SELinux (on Linux hosts) and Cisco Security Agent (on Solaris and Windows hosts) policy enforcement, the Java application components were further sandboxed by JVM security policies. Each host runs a Local Controller (LC) component that periodically checks the file system integrity, CPU, memory and disk usage of the host, and acts as the aggregator and reporter of local host based policy violation reports. The clients run three instances of LCs that monitor each other and can take over if one crashes.

The undefended JBI had only one data flow, namely the data flow associated with the PSQ (Publish, Subscribe, Query) operation. The DPASA architecture introduced four new types of data and control flows in the defended system: 1) Each software component sends heartbeats to the Core, 2) NIDS, Host Intrusion Detection System, LCs and software components all send alerts to the Core, 3) SMs send coordination and command messages among themselves and to other components, sometime via the DCs (Downstream Controllers), and 4) the PS (Policy Server) talks to ADF (Autonomic Distributed Firewall) NICs for status/heartbeats and key renewal and audit messages.

All communications between a client LAN and the Core, which were routed in the undefined JBI, are now proxied at the crumple zone except for the PS traffic. The PSQ flow is changed further. With the addition of the four PSQ servers in the four quads, a PSQ request first goes to the AP (Access Proxy), which checks whether the traffic belongs to an existing session by consulting the DC. If so, the request is forwarded to the PSQ server, which checks for signatures, and whether the client is authorized to send the request. If both succeed, the request is disseminated to the other quads. Each quad processes the request and sends a signed ACK or response back to the client. The client completes the request if it sees enough signed responses, otherwise keeps resending indefinitely. If the PSQ request is a publication, the IO (Information Object) is escrowed. When the escrow period is over, the IO is released for circulation.

4 Evaluating the Defense

We have taken a multi-pronged approach to evaluate the effectiveness of the survivability architecture. Team-members not involved in the design and development of the architecture evaluated the design and the defense-enabled system using an Integrated Survivability Validation methodology. The validation process and results are described in detail in [14]. Based on the attack trees and minimal attacks considered, they found that the defense-enabled system put up multiple independent layers of defense in front of the attacker. In particular, none of the high level goals (such as corrupting an IO (Information object), stealing an IO, or making the PSQ service unavailable) they considered had a minimal attack consisting of only a single step.

The defense-enabled system was subjected to numerous fault and intrusion injection test runs where failures and attack effects were simulated or caused by hand in various parts of the system to see how the system reacted, and whether mission operations could still continue. Apart from the external Red Team exercise (which constituted of multiple runs of the mission during which the Red Team was attacking the system with fairly unrestricted rules of engagement), the defense-enabled system was subjected to several internal Red Team experiments as well.

The findings from these evaluation activities showed that the survivability architecture significantly improves the survivability of the undefended system, even though several weak points of the architecture were discovered. Causing LOI or LOC was found to be very difficult. LOA of the PSQ service in the JBI core is also found to be very hard to cause, but the communication infrastructure, especially the communication between the LANs and the public network was found to be vulnerable to DoS attacks.

In this section we show the value of the survivability architecture by summarizing a subset of fault/intrusion injection test cases, and attack steps that the defense-enabled system was subjected to. The official results of the external Red Team exercise will be forthcoming from the independent White Team, thus this paper we will only mention some of the attack steps and the Blue Team's interpretation of events.

The sequence of attacker locations indicated by callouts in Figure 4 can be thought of as steps in a sustained attack, where the attacker starts from the public network (location 1) and progresses deeper into the Core zone. For each attack location, a description of attack step or the injected failure condition/attack effect, and how the system defends against that is described in a table format.

4.1 Public Network (location 1)

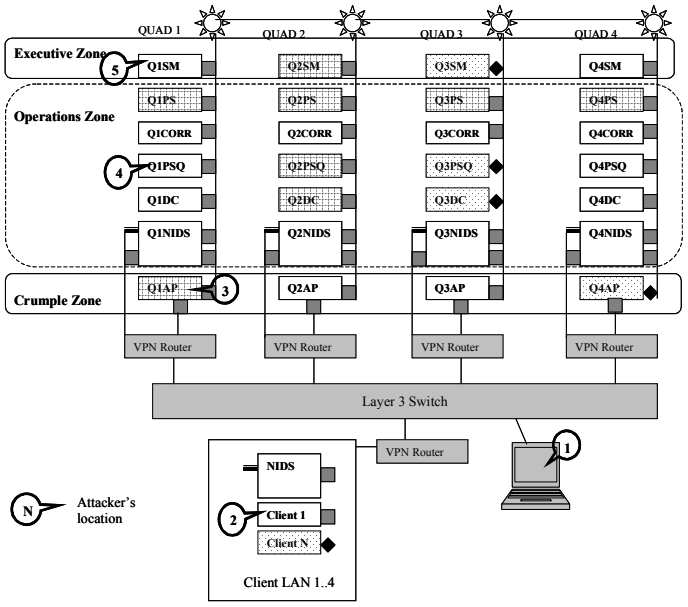

Figure 4 Attack Locations

In the first set of attacks, the attacker is connected to the Layer 3 switch via a Switched Port Analyzer (SPAN) port, a VLAN port, and an unaddressed port (which defaults to being a Trunk port[4]). Since the layer-3 switch emulates a public network like the SIPRNet in the defense-enabled system, this level of access means that the attacker has a high level of presence in the public network and can observe all traffic in it. Attempts to get inside the client or quad LANs, or observing application traffic are faced with 2 layers of VPN (VPN routers and the ADF enforced VPGs). The VPN routers and the ADF NICs are also key to stopping flood traffic from getting inside the LANs. However, we observed that the NIDSes do not pick up attack traffic that is discarded by the VPN routers because of their position behind the VPN routers. Also, since the VPN routers were a single point in the communication path from a LAN to the emulated public network, and they lacked diversity, exploitation of an implementation or configuration flaw in the VPN router proved to be fatal.

Attack Action	Defensive Action
Send traffic to a client VPN router to decrease system availability[5].	VPN Router's firewall drops non-valid VPN traffic. No reply sent back.
Sniff network traffic[6].	VPN traffic is encrypted so the attacker can only tell

[4] Trunk ports are used to connect routers, using Direct Trunk Protocol (DTP), which makes it a highly privileged port.

[5] This occurred in internal Red Team experiments as well as the external Red Team exercise.

[6] This occurred during the external Red Team exercise.

	that traffic is passing between LANs but cannot decipher the data.
Flood VPN router using valid replay traffic to disrupt the flow of heartbeat messages and IOs[7].	VPN router drops some traffic, but very high level of flood causes heartbeat and other traffic to be dropped. SM operator notices this.
Disrupt VPN tunnels by becoming a man-in-the-middle in the VPN connection[8].	All inter-LAN communication is lost.

4.2 Client Host (location 2)

In this second set of attacks, we assume that the attacker has physical access to a client host running SELinux, and has the root password. He is now inside the VPN and can interact with the members of the VPGs that the client host is part of. The attacker attempts to disrupt the mission by corrupting the IO flow. Defense against this type of attack included the adaptive response mounted by the LCs (Local Controllers) running on the client's host, modular isolation imposed by the protection domains policies, and application level security checks.

Attack Action	Defensive Action
Open an *ssh* session to an AP (Access Proxy) host which the VPG and VPN policies allow the client to communicate with.	One of the LCs running on the client host notices this and destroys the *ssh* session.
(2-1)Use an existing cmd line window and perform *ps* to get a list of running LCs.	Protection domain policies severely limit what the root user can do. This operation fails.
After (2-1), issue a *kill* command for all three LCs.	The host has been hardened, thus the *kill* command has been removed from the system. There is a special *kill* command that the LCs can use, but the protection domain policies prevent root to use.
Install own copy of the *kill* command into */usr/bin*.	SELinux policy (part of system protection domain) prohibits him from installing. As a second line of defense, the LC would remove this file if the attacker manages to install it.
(2-2) Install own copy of the *kill* command into */tmp*.	None. The LC does not remove this since the LC does not protect */tmp*.
After (2-2), issue a *kill* command for all three LCs.	LCs die[9] and stop sending heartbeat messages. SMs display a lost heartbeat message to the security operators. One of the security operators notices this and detaches the client.
Publish an IO (Information Object) that the client is not authorized to do so to disrupt the JBI mission.	Request is denied by the authorization check.
Publish an authorized IO but increase the size of the IO.	The AP in the crumple zone performs size checking and drops this IO. Or if the size were not big enough for the AP, Guardian would notice that the size is too big and send an alert to the SM (System Manager).
Publish an authorized IO multiple times.	Guardian notices that too many of the IO is coming in. Sends alerts to the SM and the SM displays the alert to the security operator. The SM operator recalls the repeated IOs.

4.3 Crumple Zone – AP Host (location 3)

In this set of attacks, the attacker has made it into the Core, and now is on the Q1AP host as the root user. The protection domain policies, adaptive responses mounted by the LC disrupt the attacker's attempt to become a man-in-the-middle using Q1AP. The policy violation reports and NIDS alerts provide visibility of attack action. Weak assumption based design of the PSQ (Publish, Subscribe, Query) protocol tolerates delay and corruption caused at this AP by retransmitting to other quads. Cross checking among PSQ servers and end-to-end signature checking are also used.

Attack Action	Defensive Action

[7] This attack was successful in causing an impact at the external Red Team exercise but the architecture did not provide any visibility of success for the attacker, so they moved on to another attack.

[8] In the external Red Team exercise a zero-day attack of this type was executed, which killed the mission. This attack required the trunk port on the layer-3 switch.

[9] This assumes that the attacker compromises the protection domain policy enforcement mechanism at this point. Otherwise, root does not have the privilege to kill DPASA processes.

(3-1) Take over a PSQ Proxy by killing the running PSQ proxy, installing his own version of the PSQ Proxy, and starting it to act as a man-in-the-middle[10].	1. Host Protection Domains prevent the killing of the PSQ proxy. If not, 2. Host Protection Domains prevent installing new PSQ Proxy. If not, 3. LC notices a change in the file system and restores it. If not, 4. LC notices a change of PID of the PSQ Proxy and kills the PSQ Proxy. All above defensive action send alerts.
(3-2) Exploit timing to kill Cisco Security Agent before detection, and then kill LC[11].	SM notices loss of heartbeat. Block the NIC to stop further attack propagation.
After (3-2) and (3-1), as a PSQ Proxy, corrupt the content of the IO.	Since an IO is signed end-to-end, the PSQ server sends an alert and rejects this IO. Client does not receive an ACK so resends the IO.
Take over a DC (Downstream Controller) Proxy. Drop the client (one that was detached earlier)'s request to connect.	1. Host protection domains and LC prevent this similar to the first row of this table. If not, 2. After inviting the client, the SM operator notices that the client is not registered. The operator checks the alerts generated by the LC and others on the Q1AP and blocks Q1AP's NIC. Client is re-invited from the Q1SM and connects successfully.
Break into Q1PSQ exploiting the same security hole	Q1PSQ runs different operating system so the same security hole does not exist, unless it is in Java VM.
Connect to SM	There is no physical connection between the executive zone and the crumple zone.

[10] This was attempted in the external Red Team exercise.

[11] When the external Red Team was allowed on Q1AP, they successfully did this and ran their own PSQ proxy. The PSQ Server noticed the change of the serial ID and dropped the traffic sent from this proxy.

4.4 Operations Zone – PSQ Host (location 4)

In this set of attacks, we assume that the attacker has made it into the operations zone, and has access to the Q1PSQ host. The redundancy of multiple PSQ servers, the PSQ server's design based on weak assumptions (where it does not solely trust its own decision but uses a quorum), detection and correlation (polling of component's status), and an adaptive response mounted by an SM counter the attacker's attempt to disrupt the PSQ flow.

Attack Action	Defensive Action
Shut down the database server running on Q1PSQ.	1. Host Protection Domain prevents this. If not, 2. Q1SM polls the status of the database, notices that it no longer responds, and displays the alert to the SM operator. Restarts the database and the PSQ operations continue.
Modify the role table in the database and delete a role for a client.	Q1PSQ gets unauthorized reply but before replying as such, consults other PSQ servers. Other PSQ servers reply authorized, so the client is not denied.

4.5 Executive Zone – SM Host (location 5)

In the final set of attacks, we assume that the attacker has made it deep into the system, past the operations zone into the executive zone and is now on the Q1SM host. The redundancy of quads (multiple SMs and PSQ Servers), adaptive responses mounted by the SMs, and the design based on the weak assumptions (where SMs do not trust each other), stops the attacker's attempt to compromise another quad.

Attack Action	Defensive Action
Detach the clients.	Other SMs notice that the clients have gone away, one of the SM re-invites them. Clients continue from where they were stopped.
Recall IOs to interrupt the JBI mission.	Q1PSQ recalls the IOs but other three PSQ servers do not. The IOs are preserved.
Issue a command to isolate Q1PSQ.	Isolating a PSQ server requires a quorum of two, no

	other SM agrees so the PSQ server is not blocked.
Issue a command to isolate Quad 3.	Isolating a Quad requires an agreement from all SMs. No other SM agrees, Quad 3 is not isolated. SM operators on other quads notice and vote to isolate Quad 1. System continues to operate with three quads.

5 Related Work

Since we first showed the feasibility of integrating defensive adaptation into an application in 1999 [8], adaptive response has become an integral part of survivable systems. Many researchers, who have explored adaptive cyber defense, have also developed special purpose architectures for intrusion detection and response (ID&R), intrusion tolerance, or graceful degradation.

The ITUA [2] project developed technology and system design techniques for building information systems that will tolerate, i.e., continue to function without violating program and data integrity, a specific class of attacks, namely, the attacks that introduce corruption in communication and application level interaction in a distributed objects application. In addition to corruption tolerant algorithms, ITUA developed an architecture for managing distributed object replicas and the hosts on which they run. The DPASA SMs and LCs are based on elements of the ITUA architecture.

The Willow architecture [9] achieves intrusion tolerance using a combination of disabling of vulnerable network elements when a threat is detected or predicted, replacing failed system elements, and reconfiguring the system if non-maskable damage occurs. Willow uses its own event-notification service as the control mechanism of its scalable architecture.

Dependable Intrusion Tolerance (DIT) [10] comprises functionally redundant HTTP COTS servers. These servers run on diverse operating systems and platforms, use hardened intrusion-tolerant proxies that mediate client requests and verify the behavior of server and other proxies, and include monitoring and alert-management components based on the EMERALD [11] Intrusion Detection System. The system adapts its configuration dynamically in response to intrusions and other faults. DIT focused on a specific kind of server (web servers), however, its use of EMERALD in sensing and alert management influenced the alert management and correlation aspect of the DPASA architecture.

Malicious and Accidental Fault Tolerance for Internet Applications (MAFTIA) [12] is a European project developing an open architecture for transactional operations on the Internet. MAFTIA models a successful attack on a security domain, leading to corruption of processes in that domain, as a fault; the architecture then exploits approaches to fault tolerance that apply regardless of whether the faults are due to accidents or malicious acts. MAFTIA is explicitly middleware-based and provides both protection from and tolerance of intrusions.

The Saber [13] system uses several mechanisms including intrusion detection, automatic code patching, process migration, and filtering of distributed denial-of-service floods for defense, but focuses primarily on server availability.

6 Conclusions

This paper presents a set of design principles we followed in designing the survivability architecture. While our experience validates the principles at a general level, we had to overcome several practical challenges while translating the theory into implementation. For instance, creating a correct and consistent policy for multiple layers of mechanisms is not trivial. We used automated policy generation to mitigate that risk partially. Finding static diversity was also challenging. Different operating systems and JVM implementations were the only source of static diversity we used. Existing system components and security tools were not always available for all operating systems, which added to the complexity. We also found that depending on how diverse entities are interconnected, more diversity is not necessarily better which is the reason we used three operating systems on four quads. On the other hand, use of weaker assumptions and the overall strategy of combining protection, detection, and adaptive reaction proved to be very useful. In particular, the PSQ protocols showed tolerance against a wide range of environmental variations and corruption (fuzzing) attacks. Human assisted responses complemented the automated responses in recovery attempts when Red Team attacks resulted in component failures. Heavyweight adaptive responses like isolating a client or a LAN had a human override option, a feature that we believe will remain useful until adaptive responses are driven by a cognitive decision making capability.

The defense-enabled JBI and its evaluation illustrate the utility of our approach of using a survivability architecture in defending an existing system. It shows that it is possible to integrate COTS and laboratory quality mechanisms, organized in multiple overlapping layers to provide a high level of resilience, without having to alter or sacrifice any of the operational features of the undefended system. Compromising the integrity or the confidentiality of the information objects proved to be very hard. The PSQ service demonstrated tolerance and graceful degradation when less than four quads were operational.

We view the success of the survivability architecture and design to be a significant step forward in the continu-

ing fight against the threat of cyber-attack. The defense-enabled JBI completed the 12-hour mission despite visible impacts caused by sustained attacks from the external Red Team. However, in another run, a Red Team was able to mount a zero-day attack on the VPN routers to stop all communication between the client and the core LANs. The evaluation of the results from the exercise is continuing, as it is a complex set of various objectives, tests, and multiple (sometimes conflicting) results. It is already clear, however, that further evaluation and continued improvement of aspects of the defense are absolutely necessary.

The experience with designing, building, and evaluating a real system under stress has provided us with quite a number of insights for future improvements in both the design and analysis of survivable systems. A few of these, focused on areas currently less well understood, include:

- Creating additional adaptive actions and survivability mechanisms, with properties that are provably uncircumventable.
- Dynamically but credibly computing and changing the trust of a component based on past actions and current systems state
- Analytically comparing the effectiveness of and contribution of the various and varying overlapping layers of defense.

Acknowledgements

The authors would like to gratefully acknowledge contributions of Bill Weinstein (Draper Laboratories), Al Valdez (SRI), Dick O'Brien and Charlie Payne (Adventium), and David Levin and Rick Schantz (BBN) for the work described in this paper. We would also like to thank Lee Badger (DARPA) and Patrick Hurley (AFRL) for their continuing support.

References

[1] M. Atighetchi, P. Pal, F. Webber, R. Schantz, C. Jones, J. Loyall. "Adaptive Cyberdefense for Survival and Intrusion Tolerance", IEEE Internet Computing, Vol. 8, No. 6, November/December 2004, pp. 25-33

[2] M. Cukier, T. Courtney, J. Lyons, H. V. Ramasamy, W. H. Sanders, M. Seri, M. Atighetchi, P. Rubel, C. Jones, F. Webber, P. Pal. R. Watro, and J. Gossett. "Providing Intrusion Tolerance With ITUA", Supplement of the 2002 International Conference on Dependable Systems and Networks, June 23-26, 2002.

[3] Tom Markham, Lynn Meredith, and Charlie Payne. "Distributed embedded firewalls with virtual private groups", Proceedings of the DARPA Information Survivability Conference and Exposition, Volume II. Washington, D.C., April 2003. DARPA, IEEE.

[4] Schneider, Fred. "Byzantine generals in action: Implementing fail-stop processors", ACM Transactions on Computer Systems, Vol. 2 (2), 1984, pp. 145-154

[5] AFRL JBI homepage: http://www.infospherics.org

[6] W. Nelson, W. Farrell, M. Atighetchi, S. Kaufman, L. Sudin, M. Shepard, and K. Theriault. "APOD Experiment 1: Final Report", BBN Technologies LLC, Technical Memorandum 1311, May 2002

[7] W. Nelson, W. Farrell, M. Atighetchi, J. Clem, L. Sudin, M. Shepard, and K. Theriault. "APOD Experiment 2: Final Report", BBN Technologies LLC, Technical Memorandum 1326, Sep, 2002

[8] Joseph Loyall, Partha Pal, Richard Schantz, and Franklin Webber. "Building Adaptive and Agile Applications Using Intrusion Detection and Response.", Proceedings of NDSS 2000, the Network and Distributed System Security Symposium, February 2-4 2000, San Diego, CA.

[9] John Knight, Dennis Heimbigner, Alexander Wolf, Antonio Carzaniga, Jonathan Hill, Premkumar Devanbu, and Michael Gertz. "The Willow Architecture: Comprehensive Survivability for Large-Scale Distributed Applications", Proc. Int'l Conf. Dependable Systems and Networks (DSN 02), supplemental vol., IEEE Press, 2002, pp. C.7.1–C.7.8.

[10] Alfonso Valdes, Magnus Almgren, Steven Cheung, Yves Deswarte, Bruno Dutertre, Joshua Levy, Hassen Saidi, Victoria Stavridou, and Tomas E. Uribe. "An Architecture for an Adaptive Intrusion Tolerant Server", Proc. Security Protocols Workshop, LNCS, Springer-Verlag, 2002.

[11] P.G. Neumann and P.A. Porras. "Experience with EMERALD to Date", 1st USENIX Workshop on Intrusion Detection and Network Monitoring Santa Clara, California, 11-12 April 1999, pp 73-80

[12] P. Verissimo, N. F. Neves, and M. Correia. "The Middleware Architecture of MAFTIA: A Blueprint," Proc. 3rd IEEE Info. Survivability Workshop, 2000.

[13]Angelos D. Keromytis, Janak Parekh, Philip N. Gross, Gail Kaiser, Vishal Misra, Jason Nieh, Dan Rubenstein, and Sal Stolfo. "A Holistic Approach to Service Survivability", Proc. ACM Workshop on Survivable and Self-Regenerative Systems, ACM Press, 2003, pp. 11-20.

[14] William H. Sanders. "DPASA Phase II Final Validation Report". Submitted to DARPA. 2005