# The DPASA Survivable JBI—A High-Water Mark in Intrusion-Tolerant Systems[1]

Partha Pal
BBN Technologies
10 Moulton Street
Cambridge, MA 02138
1-617-873-2056
ppal@bbn.com

Franklin Webber
BBN Technologies
410 W Green Street #1
Ithaca, NY 14850
1-607-877-0803
fwebber@bbn.com

Richard Schantz
BBN Technologies
10 Moulton Street
Cambridge, MA 02138
1-617-873-3550
schantz@bbn.com

## ABSTRACT
In this paper, we describe the design, development, and validation of an information system that has recently set a new high-water mark for intrusion tolerance. The system, known as the DPASA Survivable JBI, conforms to an abstract architecture for survivable systems and integrates concrete defense mechanisms for preventing intrusion and for detecting and responding to intrusions that cannot be prevented. The system has shown a high level of resistance to sustained attacks by sophisticated adversaries.

## Categories and Subject Descriptors
D.4.6 [**Security and Protection**], C.2.4 [**Distributed Systems**], K.6 [**Management of Computing and Information Systems]**

## General Terms
Design, Security, Experimentation.

## Keywords
Survivability, Intrusion-tolerance, Survivability Architecture, Defense-enabling, Defense Mechanisms

## 1. INTRODUCTION
Defending an information system against cyber-attacks is an arms race that is inherently asymmetric and favors the adversary. The adversary needs only to find a single exploit, whereas the defenders need to prevent as many of these exploits from succeeding as possible. The adversary tends to find more opportunities to attack as information systems become increasingly interconnected, and new flaws and vulnerabilities are continually being discovered (even as the defenders are addressing or coping with known ones). Information systems that are part of critical infrastructure (e.g., the national power grid, banking systems) or related to national security (e.g., weapons control, missile defense) obviously face the greatest risk.

Over the past decade, a substantial investment went to developing technologies that address specific and individual aspects of the cyber threat. For instance, firewalls focused on efficiently blocking unwanted traffic, digital signatures focused on preventing modification of data in transit, while redundancy and Byzantine protocols focused on surviving corrupt or compromised application components. Evaluation of these technologies was also limited in scope focusing only on the stated goal of the technology. While this showed good technological progress in the individual problem areas, it was unclear how to integrate these "building block" technologies in a well-defended, survivable system that tolerates sustained and sophisticated attacks.

In 2002, DARPA issued a challenge to the research community as part of the OASIS Dem/Val program, which is one of the most significant undertakings in cyber-defense research in recent days. This program sought to demonstrate, by way of developing and experimenting with a survivable DoD-relevant information system that a new high-water mark in intrusion-tolerance and survivability is achievable using the currently available technologies as building blocks. The aim was to survive against sustained and fairly unrestricted attacks from a well prepared class A Red Team for over 12 hours, whereas previous (circa 2001-2002) DARPA Red Team experiments [1] have shown survival time of a defended system only in the order of 20 minutes, with various rules of engagement restricting the attackers options. From late 2002 to early 2005, the DPASA (which stands for Designing Protection and Adaptation into a Survivability Architecture) team, led by BBN, designed a survivability architecture, used it to defense-enable the undefended target system and subjected the resulting system to multiple, fairly unrestricted Red Team exercises.

Overall, the survivable system showed excellent resiliency in thwarting, bypassing or degrading service levels for surviving the attack effects. However, as expected of a high-water mark research prototype, the survivable system needed expert operators to interpret the information it captures and to use the provided response mechanisms effectively. Nevertheless, the survivable information system produced by this program is truly a significant achievement. It demonstrated that it is possible to deploy a tightly configured, highly resilient system under aggressive cost and


*Workshop on Recent Advances in Intrusion Tolerant Systems'07*, March 23, 2007, City, State, Country.


[1] This research was funded by DARPA under AFRL contract No. F30602-02-C-0134

scheduling constraints. It also broke new grounds in validating the survivability claims of a design and its implementation.

## 2. THE UNDEFENDED SYSTEM

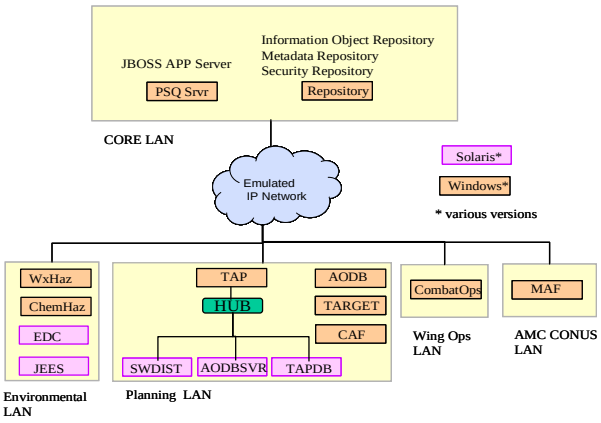


**Figure 1: The Undefended JBI**

The Joint Battlespace Infosphere (JBI) [2] concept, developed by the US Air Force Research Laboratory (AFRL), seeks to establish effective interaction among disparate military computer systems that must exchange information in support of various network-centric warfare activities ranging from intelligence gathering to mission planning and tactical operations. The JBI aims to achieve this goal by using a publish-subscribe framework that allows diverse computing systems to interact in a decoupled manner as long as they follow a common API for defining Information Objects (IO); and for performing publish, subscribe and query (PSQ) operations facilitated by a set of core services. A JBI instantiation is therefore a set of applications and core services that are needed to execute a specific mission. One such instantiation, simulating the execution of an Air Tasking Order (ATO) and planning of a concurrent airlift through the theater, was used as the demonstration vehicle in the OASIS Dem/Val program. This exemplar JBI integrates applications for selecting appropriate targets, monitoring environmental conditions, and creating ATOs. These applications are organized in 4 Local Area Networks (LANs, which are often referred to as enclaves) namely, the Planning LAN, The Environmental LAN, the Wing Operations LAN, and the AMC CONUS LAN after the Air Force functions they perform. This system is shown in Figure 1. The core services are organized in their own LAN. A successful mission would involve making the go/no-go decision on an ATO that may have WMD sites as targets. The factors influencing the go/no-go decision include presence of WMD sites in the ATO as targets, the predicted weather condition in the target area, presence of friendly force near by, possibility of other air traffic (such as the airlift mission) in the theater, etc.

## 3. HIGH-LEVEL STRATEGY

The DPASA team realized that the desired level of survivability is not achievable by any individual security tool or technique, or by a straightforward implementation of "defense in depth" — the often cited cyber-defense strategy seeking to use of multiple security techniques to mitigate the risk of one being compromised or circumvented. The following high-level strategy motivated the team to organize elements of protection, detection and adaptive response in *the survivability architecture* of the defended system:

- The architecture must create a very high barrier to entry for an attacker trying to intrude into the system, and also for the attacker who is trying to attack or move to other parts of the system after gaining access or corrupting a toehold within the system.
- The architecture must maximize the likelihood of a sensor being tripped by attacker activity by monitoring all parts of the system at all layers, perhaps in a redundant and overlapping manner.
- The architecture must support, with or without human intervention, dynamic reconfiguration of the system to recover from the damages caused by the attack, or to cope as long as possible with the attack-induced damages that cannot be repaired.

The design of the survivability architecture was further shaped by key design principles derived from decades of experience in building adaptive distributed systems integrated with defensive capabilities. A sampling of these principles appears next.

## 4. DESIGN PRINCIPLES

**SPOF protection**: It is practically impossible to eliminate (and in some cases identify) all possible single points of failure in a large and complex distributed system. But, obvious single points of failure in key parts of a system must be protected. In our case, it was clear that the PSQ server in the undefended JBI (that handles publish, subscribe and query requests from JBI clients) is absolutely necessary for continued forward progress of the mission. Therefore, the architecture ensured that no single PSQ server is an SPOF. Since we anticipated adversary trying to corrupt the PSQ service, the PSQ server was replicated 4-fold and organized in 4 *quads* (see Figure 2). With 4 PSQ servers, it was possible to tolerate Byzantine corruption in one of the PSQ servers. Spatial redundancy like server replication is not the only way to protect SPOFs. For example, the clients that issue PSQ service requests are driven by human operators, and were hard to replicate. Therefore, the architecture incorporated mechanisms to restart the clients from check-pointed state when required. Ultimately, the level of effort spent in SPOF protection is a risk-benefit tradeoff. The designer/system owners may decide that the cost of protecting a particular SPOF is too high, and assume the risk of leaving it unprotected.

**Physical barriers before key assets**: In addition to protecting the SPOFs, parts of the system that are responsible for key functionality and control decisions should have some elements of physical fortification. In our case the PSQ servers and their databases were put in the *operations zone*, which is protected by the *crumple zone*—a layer of proxy hosts that anyone seeking PSQ service must go through. The system management function (implement by the System Manager or SM), which controls many of the defense-mechanisms and adaptive responses, is put in the *executive zone*, which is situated behind the operations zone. The operations zone proxy of the system management function is known as Downstream Controller or DC. This made it very difficult for the intruder, coming from the outside of the system to attack and take control over the key decision-making and control capabilities implemented in the system management components.

**Controlled use of diversity**: Wholesale use of diversity is operationally expensive and complex. But in the DPASA architecture, we leveraged a small amount of OS diversity in the context of redundant quads and the physical layering provided by the zones in such a way that the attacker has to compromise at least two hosts running different operating systems. The general principle we recommend is that the survivability architecture should include some diversity in each access path to key assets. Apart from OS diversity, other means of introducing diversity include different processor architecture (e.g., SUN, X86), policy configuration, application and service implementations.

**Robust basis of defense in depth:** Ideally, the multiple obstacles an adversary must face in a survivable system (comprising of multiple protection measures, redundant detection mechanisms with overlapping scope and coverage, and adaptive responses) should be well-grounded, preferably upon some hardware or cryptographic base. To illustrate, note that the primary means of protecting an individual host from intrusion is to control network access to that host. A software-based firewall can do this, but then the defender and the defended would share the same hardware and OS, implying that flaws in the host and OS may affect the firewall, and the host may be attacked before the firewall can stop the attack, which in turn further weakens any other defense that assumes that the host is protected by the firewall. We reduced this risk by using the Autonomic Distributed Firewall (ADF)[3] NICs that can only be controlled by cryptographically authenticated policy servers, and provided a much stronger, separately situated and harder-to-circumvent protection than software based firewalls. Using a smart card to store and perform computation instead of storing the key on disk or using the host CPU to compute with it could provide similar advantages. The NIDS appliances and self-monitoring policy enforcement provided a similar robust basis for detection. The NIDS appliance is a hardened host that collects data, but does not respond. Policy violations, including changes in the policy or policy enforcement mechanisms are reported as detection events. For robust adaptive response we relied mostly on independent corroboration and digital signatures before triggering adaptive response; however the adaptive responses were mostly implemented in software (and some required human intervention).

**Containment layers**: Once we accept the reality that some attacks will succeed in penetrating the system, and may not be detected until their unwanted effects manifest, it becomes clear that the survivable system needs to *contain* the spread of the attack. A key architectural concept we used in this regard is known as "containment layers". Containment layers must be coordinated at the spatial level as well as the functional level. At the spatial dimension, one can think of concentric layers of increasing scope and span starting from an application process, to the host on which the process runs, to the network segment on which the host resides, and to sets of network segments and eventually spanning the entire system. However, a given functionality may be implemented by multiple processes, possibly in multiple hosts, and typically a single host runs multiple application processes. This adds the functional dimension to the containment layers. In our case, key functionality like the PSQ (i.e., the ability to address Publish, Subscribe or Query requests issued by the JBI clients) and System Management are replicated, and are made available by application level proxies that reside on the crumple zone hosts.

This view of intersecting and overlapping containment layers enabled the designers to develop key sensor and actuator mechanisms and place them in appropriate boundaries. This reinforced the overall defense by providing higher than normal access control, visibility to attacker actions and manageability of systems resources. Access control, and process and application level security policies are used to contain the attack spread from one process to other co-located processes, or from one host to others in the same network segment. Policy enforcement mechanisms, and customized sensors embedded in software applications provide indication of attempted or successful breaches of such containment boundaries. Detection or suspicion of such breaches prompt adaptive response (under defense's control) such as killing or restarting specific processes, rebooting or quarantining individual hosts, or isolating entire network segments. Note that as the container grows bigger, the level of detection and response becomes less specific—if a violation is only detected between network segments, accurate information about the compromised processes, or hosts may not be available.

**Range of adaptive responses**: It follows from the discussion of containment layers that the survivability architecture will exhibit a range of adaptive responses with varying scope and consequences. Clearly, they should not be treated uniformly—some could be mounted as knee-jerk responses, others should be mounted with deliberation. The designers should consider carefully the kind of decision-making and control mechanism to put between the various detectors and actuators introduced by earlier considerations. In DPASA we followed the simple rule of thumb: actions that have localized impact or can be easily reversed are mounted in a knee-jerk manner, typically driven by an if-then condition or a local table look-up. Examples include restoring a lost or corrupt file, re-sending a publish request, killing a spurious (i.e., that was not part of, or a descendant of the application processes) process on a local host, or blocking a request that is against the predefined security policy. However, killing an application process, or isolating a host or a network segment can adversely affect the system's desired functionality. Thus, such responses need to be coordinated. In DPASA, this coordination had two distinct parts: deciding when to mount such a response, and adapting the rest of the system to deal with the effect of such a response. For the former, our approach was fairly conservative—multiple independent corroborations were required for mounting such actions. For the latter, specialized protocols were developed to propagate or inform the affected components of the system.

**Configuration generation from specs**: Misconfiguration is often cited as the main source of security vulnerability. Use of redundancy and multiple layers of defense create more opportunity for misconfiguration. Not only policies governing the defense mechanisms must be consistent across different layers of the defense, they must be coordinated across the system among the redundant and interacting components. This risk must not be left unmanaged. In DPASA, we took the position that configurations must be auto-generated as much as possible from a single specification. We were successful in removing one key source of misconfiguration by specifying the IP address and port assignments in a file, and using that to generate all policy and configuration that needed them. Even though some policies needed to be generated manually, sourcing the IP and port assignments from a single place provided significant time and cost savings in terms of testing and debugging that was avoided

## 5. THE DEFENDED SYSTEM

The shortest way to describe the defended system is to highlight the enhancements resulting from the survivability architecture.

**System Level Enhancements:** Instead of a single JBI core enclave, the defended system has four core enclaves (quads). Each enclave runs the same functions, but consists of hosts running 3 different operating systems (Security Enhanced Linux, Windows, and Solaris). The crumple, operations, and executive zones are also depicted in Figure 2: the crumple zone has an AP host and the executive zone has an SM host in every quad. The operations zone has 5 hosts (namely, the NIDS, the PSQ, the PS, the CORR, and the DC) in each quad.

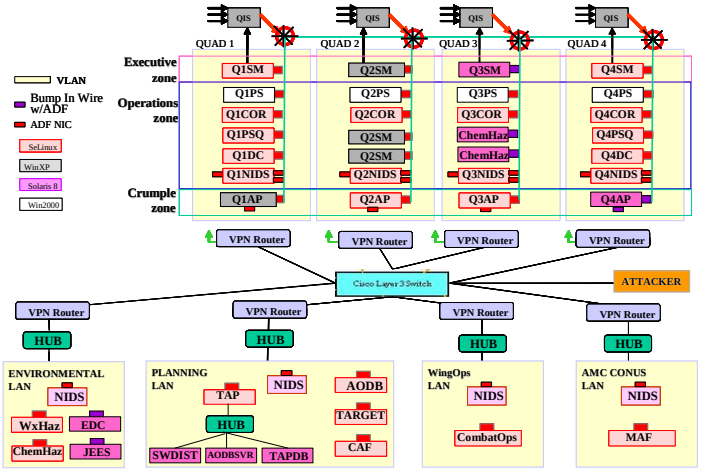


**Figure 2: The Defended System**

The NIDS is a network-based intrusion detection appliance, responsible for watching over network traffic within, coming into and going out of the enclave in which it resides. The PSQ host runs the main functions of the core part of the undefended system (i.e, the core JBI services including the JBoss server and the MySQL repositories). The PS host runs the policy server process that manages the ADF NICs in the system, and interfaces with the system management function in the executive zone for mounting adaptive responses involving the ADF NICs. The CORR host acts as an alert sink and runs two applications namely, the EMERALD correlator and the Advisor. The former consumes the alerts to correlate, ranks the aggregated alerts, and presents the ranked alerts in a GUI. The latter consumes the alerts to assess whether a given component can be trusted to remain a part of the system, and advises the system management function in the executive zone. It does so by interpreting an alert as an accusation made by a component about another; and considering factors such as whether the accuser has physical or logical connection or relation with the accused, whether or not the accused is accused by others, whether the accuser itself can be trusted, etc. The DC host runs the Downstream Controller (DC) application that acts as an intermediary avoiding a direct connection between the crumple and executive zones. In addition, the DC processes the heartbeats.

The AP host in the crumple zone acts as the port of entry for all traffic into a quad from the clients. For each process running in the operations zone (i.e., processes in the DC, PS, CORR and PSQ hosts) that the clients interact with, there is a corresponding proxy on the AP host performing rate-limiting and certain application-level checks (without violating the end-to-end authentication and encryption) before forwarding traffic to the core.

The SM host in the executive zone runs the System Manage (SM) application that manages defense mechanisms in clients and in its own quad, and also interfaces with the DPASA security operators.

The configuration of a client enclave is not changed except for the addition of a NIDS. The Client hosts run applications that make use of the mediated service in the operations zone. In the context of the JBI exemplar, these are applications that have specific roles in the mission. They are known as JBI clients, and interact with each other via publish, subscribe and query.

**Network Level Enhancements:** The defense-enabled system has a Virtual Private Network (VPN) router in front of each enclave, establishing a VPN trunk from a client enclave to each of the core enclaves. All hosts are equipped with ADF NICs, as shown by the small red rectangles. Hosts in the crumple zone have two ADF NICs since each crumple zone host has one client-facing and one core-facing interface. We worked around the lack of Solaris ADF device drivers by using ADF equipped Linux hosts as "bumps in the wire" (indicated by the small purple rectangle attached to a Solaris hosts). Each core enclave has a managed switch (HP Procurve 2524) controlling the communication between quads as well as inter-zone communication within its own quad. Each such switch is further controlled via a Quad Isolation Switch (QIS) by the 4 SMs. A Cisco 3750 was used to emulate the public IP network connecting client and core enclaves implemented as VLANs off the Cisco 3750.

**Host Level Enhancements:** In each host protection domain policies tightly define how a process can be started and limit it's the capabilities. In addition to SELinux (Linux hosts) and Cisco Security Agent (Solaris and Windows hosts) policy enforcement, the Java application components were further sandboxed by JVM security policies. Each host runs a component (known as LC) 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 violations. The clients run 3 instances of LCs that monitor each other and can take over if one crashes.

**New Dataflow:** The undefended JBI had only one data flow (i.e., the PSQ operation). The DPASA architecture introduced 4 new types of dataflow in the defended system: 1) Heartbeats sent by each software component to the core, 2) Alerts from NIDS, HIDS, LCs and software components, 3) Coordination and command messages sent by SMs among themselves and to other components, sometime via the DCs, and 4) the status, key renewal and audit messages from the PS to ADF NICs.

With the addition of the 4 PSQ servers the new PSQ dataflow is as follows. A PSQ request first goes to the AP, and then the AP checks whether the traffic belongs to an existing session by consulting the SM via DC. If this succeeds, the request is forwarded to the PSQ server, which checks for signatures and permissions. If this succeeds, the request is disseminated to the other quads. Each quad processes the request and sends the appropriate signed ACK or response back to the client. The client completes the request if enough signed responses are received; otherwise it keeps resending the request (indefinitely). If the PSQ request is a publication, the IO is escrowed. When the escrow period is over, the IO is released for circulation. The PSQ dataflow, encapsulated within a DPASA JBI middleware implementation, uses sockets over the network and JMS between the server-side end-point and JBoss.

**New Protocols:** Key protocols introduced by the survivability architecture are described below.

The **registration protocol**, used when a client is ready to join the JBI, involves the SM and the client mutually authenticating each other, and is initiated by a human operator at the SM. The **alerts protocol** used for reporting suspected events works as follows. Alerts generated in a quad are sent to a "Tee" process on the CORR host of its own quad, which passes a copy of each alert to Emerald and the Advisor. Alerts generated by clients however go to each quad (through the Correlator Proxies). The **PSQ protocol** implements the basic PSQ functions used by all JBI applications. The PSQ protocol is fault-tolerant. When all four quads are participating in the protocol, JBI clients see correct behavior from the core even if any one of the PSQ servers is corrupt and behaving in an arbitrarily malicious way. If fewer servers are participating, the protocol may not tolerate corruption but it will tolerate crashes if at least two servers participate. Corrupt access proxies are tolerated as long as one proxy works correctly. Corrupt client behavior is tolerated in some cases, detected in others. A design goal of the PSQ protocol was to isolate it as much as possible to allow other domain specific service providers to be plugged into DPASA. The **TS protocol** implements a distributed fault-tolerant clock for use by both JBI clients and DPASA defenses. The 4 TS servers respond to requests for the current time. The protocol maintains a system-wide time despite corruption of any one of the TS servers. The **heartbeat protocol** is used to detect failures in the DPASA survivability components. The heartbeat messages are sent to the core where the SM on each quad uses this to display the status of the system. The SM participates in a number of protocols, key among the **SM protocols** are the SM-PS protocol to retrieve data and exert control on ADF NICs; the PSQAdmin protocol for administrative functions on the PSQ server local to the SM, i.e. quorum group management; and the PSQDB protocol for accessing the internal state of the PSQ's database, and recalling of IOs.

## 6. EVALUATION RESULTS

In the first round of red team exercises, conducted at the Air Force Research Laboratory (AFRL), Rome, NY (March 15-18, 2005), the survivable system was subjected to two separate 12 hour exercises. Two red teams launched a number of planned and ad hoc attacks. While the first red team caused a system slow down, the critical mission objectives were met. The second red team succeeded in disrupting the communication paths between clients provided by the commercial VPN routers but could not penetrate beyond that outer layer. The success enjoyed the red teams came at the expense of commercial products deployed at the periphery of the survivable system, and were limited to disrupting the availability of inter-enclave communication. When individual defense mechanisms were directly tested, the red teams were unable to compromise any mission requirement. The results show that the survivable JBI made it very hard for the attacker to penetrate into the system, or to cause significant damage inside it.

The second round (Nov 7-18, 2005) of adversarial testing was designed to evaluate the system's resilience when the high barrier to entry at the periphery is removed and the internal architecture and defense-mechanisms are attacked directly. The red team was augmented with developers and given total access to the system including source code, configurations, keys and passwords. The augmented red team considered a wide range of attack possibilities to violate confidentiality, integrity or availability of the system. Many of these possibilities were nullified by the design of the system, leaving a set of ~24 attack ideas, of which 19 were executed. In all the attack runs the red team was given one or more hosts within the survivable system to preposition attack code, and to launch their attacks. Of the 19 runs, the red team was able to stop the mission from completion within the stipulated time in only 4. In one of these 4 runs, the survivable system recovered to such an extent that it was able to complete its mission just 20 minutes after the stipulated time. Both automatic and human assisted responses occurred in 17 out of 19 runs. Some indication of attacker activity was reported automatically in 15 runs. No operator assistance was required in 3 runs

Overall, the distributed and embedded sensors provided excellent localization of compromises. Because of the multiple layers of cryptographic and agreement protocols the survivable system did not experience any compromise in confidentiality and integrity even with high level of access inside the system and prepositioned attack code. Even attacks that aimed to cause loss of confidentiality and integrity resulted in loss of availability only. The containment layers and physical barriers in the architecture limited the adversary's visibility into the system, and thereby limiting his ability to control it. The redundancy based protocol showed a very high level of resiliency—as long as one of the redundant components was providing some level of service, the system gracefully degraded and continued to operate.

## 7. CONCLUSION

As expected of a high-water mark research prototype, the survivable system needed expert operators to interpret the information it captured and to effectively mount some responses. Nevertheless, the DPASA survivable JBI is truly a pathfinder. It demonstrated that a tightly configured system with multiple and overlapping layers of defense can be designed and implemented under aggressive cost and scheduling constraints. It also broke new ground in validating the survivability properties of a system. By demonstrating that the red teams can be successfully challenged and forced to attack the system through the legitimate entry points, it provides a renewed level of confidence in the continued fight against the cyber-threat.

## 8. ACKNOWLEDGEMENT

The authors would like to thank Lee Badger of DARPA and Pat Hurley of AFRL for their support and valuable feedback.

## 9. REFERENCES

[1] Nelson, W., Farrel, W., Atighetchi, M., Clem, J., Shepard, M., and Theriault, K. APOD Experiment 2: Final Report. *BBN Technologies LLC, Technical Memorandum 1326* (Sep. 2002).

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

[3] Markham, T., Meredith, L., and Payne, C. Distributed embedded firewalls with virtual private groups. In *Proceedings of the DARPA Information Survivability Conference and Exposition, Volume II* (Washington DC, April, 2003). IEEE