

Self-Regenerative Systems (SRS) Program Abstract

May 26, 2004

Lee Badger

This document presents the goals of the Self-Regenerative Systems (SRS) program, its anticipated schedule, programmatic details, and short summaries of the projects in the words of the SRS researchers.

BAA 03-44 released Sept. 29, 2003
Initial BAA closing: Nov. 26, 2003
Duration: 18 months
Selected performers: 11
Kickoff: July 20-21, 2004

Table 1 SRS Program Startup Details

The goal of the SRS program is to develop technology for building military computing systems that provide critical functionality at all times, in spite of damage caused by unintentional errors or attacks. All current systems suffer eventual failure due to the accumulated effects of errors or attacks. The SRS program aims to develop technologies enabling military systems to learn, regenerate themselves, and automatically improve their ability to deliver critical services. If successful, self-regenerative systems will show a positive trend in reliability, actually exceeding initial operating capability and approaching a theoretical optimal performance level over long time intervals.

To achieve the SRS program goals, the program will address four key technology areas: 1) Biologically-Inspired Diversity, 2) Cognitive Immunity and Regeneration, 3) Granular, Scalable Redundancy, and 4) Reasoning About Insider Threats.

The SRS program defines quantitative goals for each of these areas:

Biologically-Inspired Diversity: *Metric:* automatically produce 100 diverse but functionally equivalent versions of a software component such that no more than thirty-three versions of a component share the same deficiency.

Cognitive Immunity and Regeneration: *Metric:* accurately diagnose 10% of the root causes of system problems and take effective corrective action in half of those diagnoses.

Granular, Scalable Redundancy: *Metric:* attain a three-fold reduction in latency for achieving consistency of replicated data while tolerating up to five Byzantine failures in a centralized server setting and attain a fifteen-fold reduction in latency for achieving consistent values of data shared among from one hundred to ten thousand participants using epidemic algorithms, where all participants can send and receive events.

Reasoning About Insider Threats: *Metric:* thwart or delay 10% of insider attacker goals.

Performers will be responsible for assessing and measuring their own progress against these goals and reporting the results to DARPA. In the Granular, Scalable Redundancy

area, this will include a testbed and baseline measurement at the first PI meeting. To assist with this assessment, the program will include an Independent Evaluation Team (IET). The IET will be comprised of subject matter experts in the four SRS technical areas. The role of the IET will be to:

- Provide technical feedback to performers at PI meetings
- Attend site visits for in-depth reviews
- Review performer self-assessment strategies

IET membership so far:

Fred Schneider (Cornell)

Gregg Tally (McAfee Labs)

John McHugh (CMU)

Crispin Cowan (Crispin Cowan Security Consulting)

Stephanie Forrest (University of New Mexico)

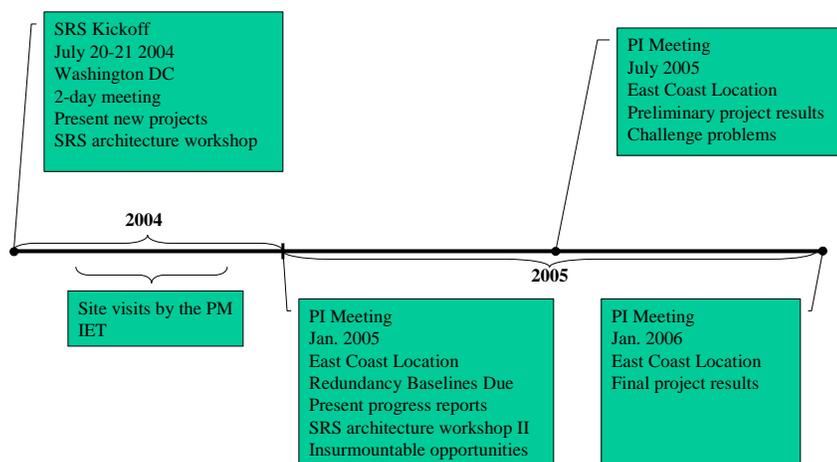


Figure 1 Preliminary SRS Roadmap

1. Biologically-Inspired Diversity

Conventional software systems are highly homogeneous; as a consequence, a single flaw can be exploited by an adversary to cause massive damage throughout a military system. In this technical area, the program will reduce the leverage available to adversaries by generating many variants of a system component that perform the same desired functions but are sufficiently different in their vulnerabilities so that a single attack can only damage a small part of an entire system. Using this strategy, the program aims to reduce the impact of any given attack and enable practical automatic recovery. If successful, the strategy will multiply the attacker work factor.

1.1. Genesis

University of Virginia: J. C. Knight, J. W. Davidson, D. Evans, A. Nguyen-Tuong

Carnegie Mellon University: C. Wang

In the human genome, single nucleotide polymorphisms—single base differences in our DNA— are believed to underlie our susceptibility to a host of diseases and our responses to various treatments. We seek to reproduce the genetic diversity found in Nature by deliberately and systematically introducing diversity in software components. The hope is that while the phenotype of software components will be similar (its functional behavior), its genotype will contain enough variations to protect the population against a broad class of diseases (attacks, aging).

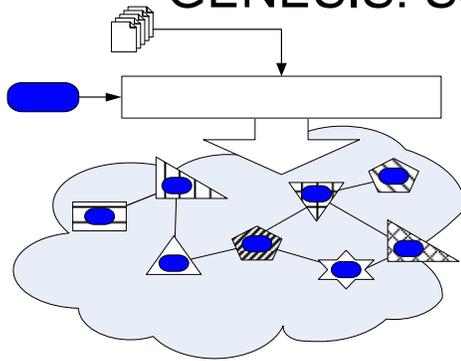
Similarly to Nature’s systematic use of evolution as its primary engine of diversity, we will use a systematic and comprehensive methodology based on two fundamental and complementary approaches, *design* diversity and *data* diversity, as our engine of software diversity.

Design diversity is the creation of multiple implementations of a given specification such that the different implementations have different designs. Data diversity is the use of multiple copies of a single implementation with each copy operating on different input data but yielding the same desired results. In data diversity, the different data streams are produced by a process known as data re-expression. Each diversity approach will be applied systematically at multiple levels of software representation to produce a spectrum of techniques for the creation of diverse software components.

An innovative aspect of our work is the adoption of recent developments in compiler, compiler-compiler and virtual machine technologies to effect both kinds of diversity transformations. The results of applying design and data diversity techniques in combination constitute *Hierarchic Multi-Diversity*—our comprehensive approach for achieving the requisite quantitative and qualitative levels of diversification.

If we are successful, our research will result in a set of techniques and associated tools that permit the creation of a large population of functionally-equivalent diversified versions of components such that a significant fraction of the population does not share the same vulnerabilities.

GENESIS: Software Diversity



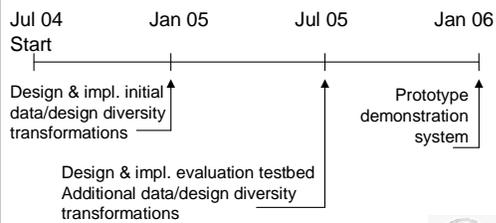
New Ideas

- Comprehensive application of diversity
 - Design & data diversity
 - Dynamic diversity
- Hierarchic approach
 - Transformations at multiple levels of representations
- Automation
 - Meta-compiler & compiler technologies
- Dynamic Virtual Machine
 - Apply diversity at run-time via virtual machine technology

Impact

- Demonstrate feasibility of large-scale production of functionally-equivalent software variants
- Reduced susceptibility of software population to cyber threats
- Testbed for evaluation of diversity techniques
- Demonstrate feasibility of dynamic diversity technology on COTS

Schedule



University of Virginia: J. C. Knight, J. W. Davidson, D. Evans, A. Nguyen-Tuong
Carnegie Mellon University: C. Wang

Diversity Specifications



1.2. DAWSON -- Synthetic Diversity for Intrusion Tolerance

Global Infotek, Inc.: James Just

Software

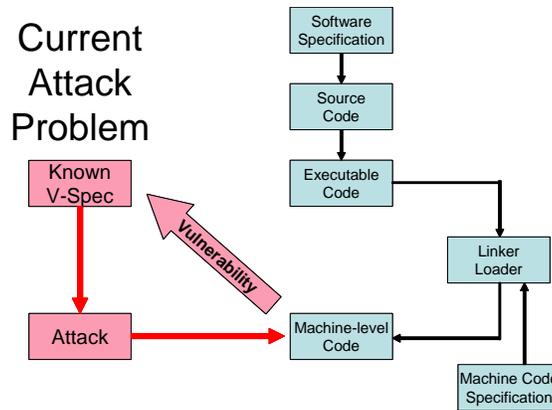
Affordable, robust systems that respond automatically to accidental and deliberate faults are very desirable. The current state of the art combines fault- and intrusion-tolerance technologies to produce robust, survivable systems. Such systems have an Achilles heel. Their robust performance depends upon the continued existence of spare resources for failover. Spare resources can be depleted by continued attacks until the system can no longer maintain critical functionality.

The Diversity Algorithms for Worrysome Software and Networks (DAWSON) project will develop prototype software that mitigates these problems for commercial off-the-shelf (COTS) software through synthetic diversity introduced at the binary code level. DAWSON's approach randomizes critical information in the binary to alter the interface and representation conventions in such a way that injected code no longer functions and existing code is no longer reachable.

Breaking Vulnerability Specifications

The author of any computer attack focuses on specific vulnerability details such as specific branching address locations and how to exploit them to point to his injected code. The injected code must find and execute system calls to access system resources, talk over the network, propagate further, etc. Such details can be viewed as a Vulnerability Specification or V-SPEC from the perspective of the attacker. Any program that supports the assumptions of the V-SPEC is vulnerable to that attack. This is illustrated below.

Diverse popul
equiva

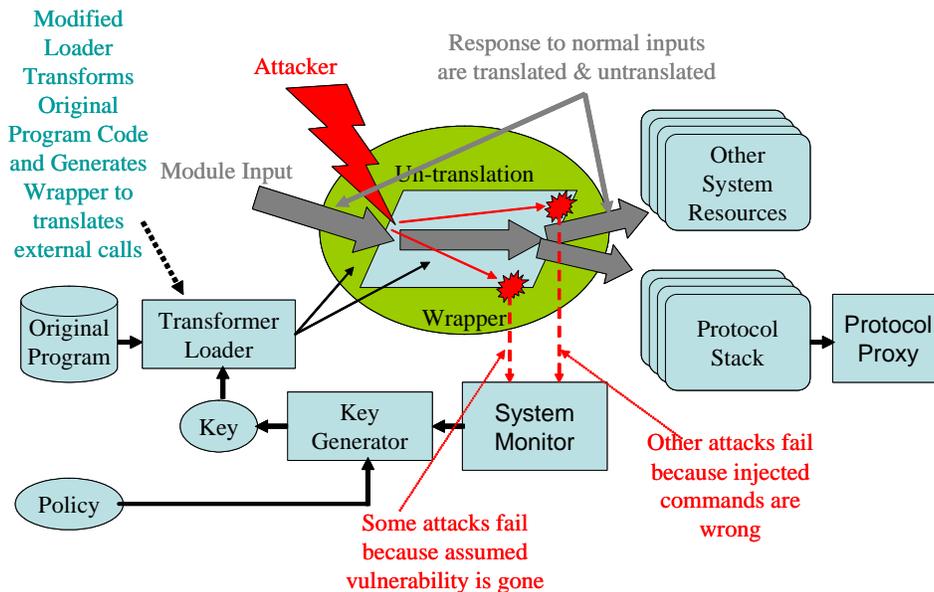


Our approach transforms programs in ways that break the V-SPEC, without affecting legitimate assumptions about program behavior (the A-SPEC). For example, if random sized blocks of information are pushed onto the stack to make stack locations harder to predict, legitimate programs be affected by such things.

DAWSON Architecture

There are some transforms that are easy to perform on executable code with just the loader. Others take more or less analysis at the binary or disassembled level. Still others can only be accomplished if the source compiler has provided hints or annotations about the executable. The DAWSON system architecture accommodates these differences and is illustrated below.

Diversity System Functional Architecture



Not shown in the above diagram is an offline analyzer above preprocesses the original program code, e.g., uses linkage information in PE formatted executables and DLLs to locate entry points, unresolved absolute addresses, and system call linkages. It then generates an annotation file for each program. The Transformer-Loader module uses

these annotations, a key and policy to modify the loading program's memory layout and to build any needed wrappers to retranslate external calls.

The transformed program executes on the same hardware but has distinctly different assumptions about its resources and interacting with its environment. Each load is changed unpredictably based upon a random key.

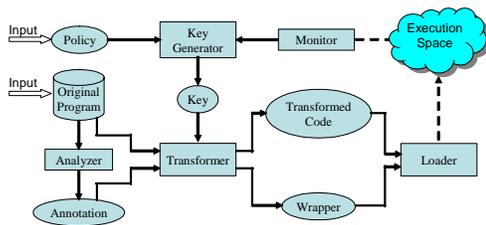
Most attacks will fail because the assumed vulnerability location is different. Others will fail because the injected commands do not find the system resource names they need. With high probability, attacking code will simply fail and crash the process or endless loop which increases the detection likelihood.

Research Products

The DAWSON project will deliver automated runtime diversity for software at the level of component libraries, operating system calls, and protocol stacks for COTS computer systems. Our implementation will focus on the world's largest monoculture (Microsoft Windows, Intel hardware, and Office/BackOffice applications) but the approach and techniques developed will be applicable across the spectrum of other platforms.

Diversity Algorithms for Worrisome Software and Networks (DAWSON)

DAWSON System Architecture Overview



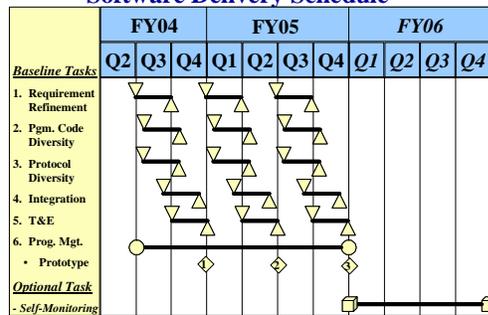
Technical Approach

- DAWSON will use randomized runtime transformation of executable code
- Runtime transformation of binary code and machine addresses and state space expansion techniques
- Random cryptographic keys will rejuvenate the code with unique transformations on each restart.
- Policies can determine which type of transform is favored.

Impact

- We will introduce spatial and temporal diversity to common Windows applications, the largest computer monoculture in the world.
- This current baseline of software and network protocols turns over very slowly and constitutes the bulk of security vulnerabilities on extant Internet and military systems.
- DAWSON is a key enabler of fourth generation, secure, survivable systems and will be easily integrated with other SRS components.

Software Delivery Schedule



2. Cognitive Immunity and Self-Healing

Although some fault-tolerant systems perform automated error masking and recovery for benign faults, no similar and effective capability exists for tolerating and recovering from malicious attacks. In this technical area, the program will develop techniques that introspect about a systems operation, that recognize damage resulting from successful attack, and that reason about appropriate countermeasures. Through such techniques, the

program aims to build systems that automatically recover and regenerate their operational capability, even in the face of continuing attack.

2.1. Learning and Repair Techniques for Self-Healing Systems

MIT: Michael Ernst and Martin Rinard

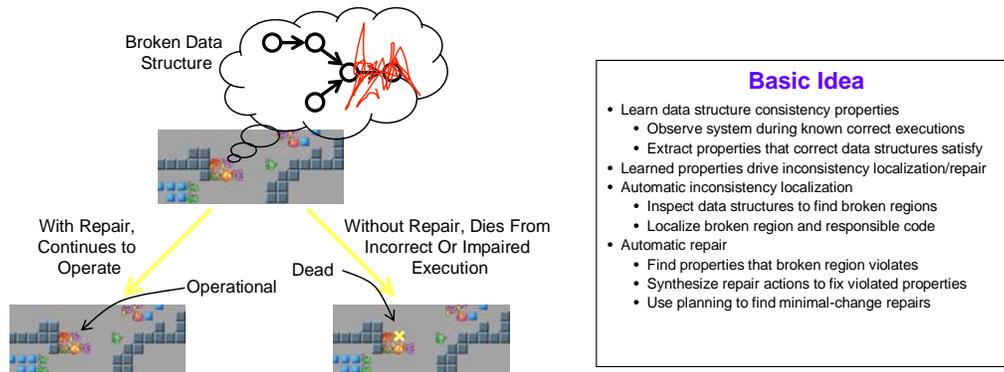
Today, computer systems remain brittle in the face of failures. Unlike biological organisms, which can respond robustly to damage from internal errors and external attacks by healing the damage and continuing to operate, computer systems lack mechanisms that allow them to detect and repair damage and avoid similar problems in the future. We propose to attack this problems with several technologies: an error localization system that uses consistency constraints to localize errors; a repair system that repairs the damage that the errors have caused; a learning system that dynamically inspects the operation of the system to automatically learn important consistency properties (these properties will then be used to drive the error localization and repair system); an upgrade evaluation system that projects the impact of potential software upgrades on the system (enabling it to reject potentially harmful upgrades and integrate useful upgrades); and an automatic mode selection mechanism that matches the operating mode of the system to the demands of the environment in which it finds itself (thus enabling the system to better survive unpredictable combinations of attacks, damage, and environments).

Deliverables

The deliverables for this project fall into three broad categories. First are a set of common tools and languages that will be used throughout the project, tying together the research thrusts and enabling them to work together synergistically. Second is a set of specialized tools that build on this common foundation to achieve each research objective, together with additional integration of these tools. Third is a set of experimental results that evaluate both the novel techniques that we propose, and also our prototype implementations; these experiments will point the way to future research and practical application. The foundational tools include: (1) a common consistency specification language that will be used by all the tools and that is capable of expressing correctness requirements of software system, (2) a prototype implementation of an algorithm that detections violations of the constraints, and (3) a system for automatically inferring constraints (a form of specification) for an arbitrary program. Together, these tools enable use of arbitrary code in our experiments without requiring humans to perform tedious, error-prone, and ad hoc specification tasks. The specialized tools include prototype implementations of algorithms that: (1) localize information representation corruption errors to specific regions of code, (2) automatically repair corruption in a manner consistent with all other constraints, (3) evaluate whether a particular upgrade is likely to cause errors if integrated into a particular system, and (4) select an appropriate operating mode for a system with such modes. The integration efforts will combine (1) the learning, localization, and repair tools to automatically create self-healing systems that recover from certain corruption errors, (2) the localization and upgrade evaluation tools to predict impact of upgrades on specific parts of a codebase, in particular for correcting specific existing errors, and (3) the localization and mode selection tools to select different operating modes that work around errors in specific modes. The

experimental results will evaluate each of the above tools and integrated systems on a variety of systems, including the CTAS air-traffic control system (deployed in 7 of 21 air-traffic control centers in the continental United States), the Linux C standard library, competition robots from MIT, free software such as the Freeciv interactive game, and other software systems.

Learning and Repair Techniques for Self-Healing Systems



Basic Idea

- Learn data structure consistency properties
 - Observe system during known correct executions
 - Extract properties that correct data structures satisfy
- Learned properties drive inconsistency localization/repair
- Automatic inconsistency localization
 - Inspect data structures to find broken regions
 - Localize broken region and responsible code
- Automatic repair
 - Find properties that broken region violates
 - Synthesize repair actions to fix violated properties
 - Use planning to find minimal-change repairs

Impact

- Broken data structures have many causes
 - External attacks, Internal errors
 - Unexpected operating conditions
- Broken data structures can have severe consequences
 - Catastrophic failure
 - Security breaches and compromised systems
- Our techniques promise to
 - Enable detection and healing of data structure damage
 - Make systems more robust, reliable, and secure
 - Help systems continue to operate successfully through attacks, failures, and errors

Accomplishments and Status

- Program not yet started
- Components in various stages of development
 - Data structure consistency property learning
 - Inconsistency localization
 - Inconsistency repair

Martin Rinard, Michael Ernst, MIT Laboratory for Computer Science

2.2. Pervasive Self-Regeneration through Concurrent Model-Based Execution

MIT: Brian Williams, Gregory T. Sullivan

1. Pervasive system robustness by composing concurrent fault aware processes.

In open systems, failures can occur within *any* subsystem, process or component of the system, not just at its perimeter. To achieve robustness for open systems, we will enable *every* process to be *fault aware*, by recognizing and adapting to failure. In contrast to traditional, centralized approaches, our approach will support fault-aware processes that operate concurrently while communicating across a network, and that operate through a layered architecture within a single process.

2. Fault-adaptive processes through model-based program execution.

To achieve robustness pervasively, fault adaptive processes must be created with minimal programming overhead. *Model-based programming* elevates this task to the specification of the intended state evolutions of each process. A *model-based executive* automatically synthesizes fault adaptive processes for achieving these state evolutions by reasoning from models of correct and faulty behavior of supporting service components. This synthesis includes methods for transitioning to intended

states, monitoring progress, diagnosing failure, and repairing or reconfiguring underlying components. Furthermore, the model-based executive can construct *novel recovery actions* in the face of *novel faults*.

3. Self-deprecating methods through prognostic mode estimation.

As with traditional languages, model-based programs are specified in terms of a set of methods and method invocations. Execution of these methods will fail if the service components they rely upon irreparably fail. Model-based execution will enhance robustness by continuously deprecating any method that is involved in the current execution plan and whose successful execution relies upon an irreparable component.

4. Self-regenerating methods through redundant method dispatch.

When a method is deprecated, the model-based executive will attempt to regenerate the lost function that caused the method deprecation by reasoning about its service component models in order to repair or reconfigure the faulty services. To handle the event of permanent method deprecation, a model-based program includes a specification of redundant methods for achieving each desired function. If a deprecated method cannot be repaired, the desired functionality is regenerated dynamically by choosing a suitable redundant method and verifying correct function.

5. Self-optimizing methods through decision-theoretic dispatch.

In addition to failure, component performance can degrade dramatically, reducing system performance to unacceptable levels. To maintain optimal performance, decision-theoretic method dispatch will continuously monitor performance, and select the currently optimal available method that achieves the desired function.

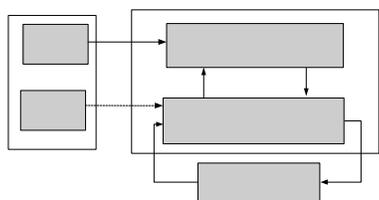
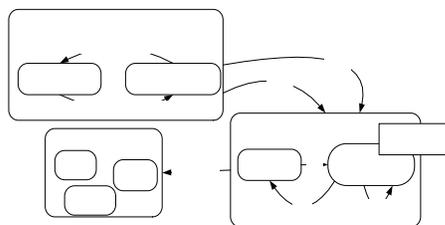
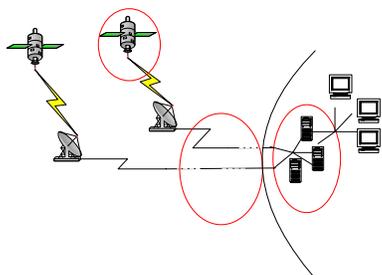
6. Safe fault adaptation through method dispatch as continuous planning

Failures can occur at any moment and can have a disastrous effect if not immediately caught. Our approach will continuously monitor failure and performance degradation, and will reactively invoke its regeneration processes as required.

7. Incorporation of fault adaptation incrementally.

Improving robustness of large, legacy systems must be a gradual process, whose cost can be amortized over time. *Our approach allows us to add robustness to individual software components, thus incrementally increasing the robustness of the overall system.*

Pervasive Self-Regeneration through Concurrent Model-Based Execution



- **Concurrent Model-based executives:** monitor, predict, diagnose, replan, to keep software systems running and achieving goals.
- **Model-based Programming:** high-level, probabilistic, goal-driven specification of intended, nominal, and off-nominal behavior.
- **Decision-theoretic dispatch:** select from redundant methods based on reliability, performance, reward.
- **Self-deprecation and repair:** deprecate methods based on failed dependencies. Schedule repair based on value.

MIT CSAIL, Brian Williams, DARPA SRS, May 2004

2.3. Architectural Differencing, Wrappers, Diagnosis, Recovery, Adaptivity, and Trust Modeling (AWDRAT)

MIT: *Howie Shrobe*

Satellite

Teknowledge: *Bob Balzer*

We are building an infrastructure, named AWDRAT, that helps software systems respond in reasonable ways to compromises of the resources, avoiding them if the compromise would cause serious harm, but using them in pursuit of important goals if they can be employed without fear of damaging properties of interest.

Satellite dish

Software hosted within the AWDRAT environment must be structured as variant methods capable of rendering common services, allowing AWDRAT the freedom to select the best method for achieving the task. When there are multiple resources that are more or less equivalent, AWDRAT has the freedom to choose between them. The AWDRAT decision cycle is:

- When presented with a task to be achieved, AWDRAT consults its method library, finding all applicable methods relevant to the service request. Each combination of method and supporting resources is evaluated, taking into account the cost of the resources and the value of the service quality delivered. AWDRAT's trust model is also consulted to assess the possibility that the resources are compromised and to include the cost of a potential failure. The method and set of resources that promise the best overall tradeoff is selected for execution.

Network

- Accompanying each method is an architectural model of the computation performed by the method. AWD RAT interprets this in parallel with the executing code, using wrappers to extract data from the method's execution and noting when the executing code violates a constraint of the architectural model. This technique is called Architectural Differencing. If any constraint imposed by the architectural model is violated, model-based diagnosis is invoked to assess what part of the computation may have failed and the trust model is updated with the information produced by the diagnosis, leading to new assessments of the trustability of the computational resources.

- Recoverable data (e.g. databases, code segments, password files) are restored in order to establish a consistent point from which to resume the computation. AWD RAT then returns to the beginning of its decision cycle, informed by the results of diagnosis. It restarts the computation from a place guaranteed to have been successfully completed and chooses a new method and set of resources in light of the updated trust model.

This approach guarantees that AWD RAT will find some way to achieve the application's goals if there is an available method; it also guarantees that it will steer the application clear of resources that it has reason to believe are corrupted if the compromise to the resources is likely to cause damage. The application system behaves adaptively when hosted within the AWD RAT environment.

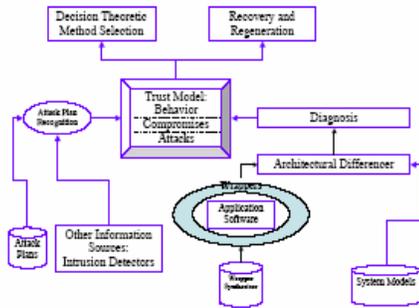
Deliverables

We will develop and deliver a prototype of the AWD RAT environment that includes the following:

- A trust model that assesses the likelihood that each computational resource is compromised in a specific way and thereby implies for what purposes the component may be used reliably.
- An infrastructure to support self-adaptive application systems. Such systems will dynamically decide how best to achieve each goal in light of current conditions, in particular, in light of the trust model.
- A system modeling framework that allows AWD RAT to operationalize the specifications of an application in terms of conditions expected to hold at particular points and invariants that must be true across intervals of the computation.
- A synthesis system that automatically generates wrappers such that the important state of the computation is observable and such that redundant copies of critical data can be automatically provisioned.
- A diagnostic component that is activated by the detection of a symptom (i.e. the failure of a computational component to behave in accordance with its specification) by some wrapper. The diagnostic component then determines what failures may have led to the observed symptom, whether this failure is indicative of a compromised resource, what the cause of this compromise is likely to have been. The diagnostic component updates the trust model to reflect its conclusions.
- A recovery component that operates after the diagnostic component has assessed the cause of the failure. The recovery component is responsible for restoring corrupted data sets to a usable consistent state and for the selecting of a suitable method for achieving the application's goals, given the updated beliefs in the trust model.

- An attack modeling capability that discovers, models and recognizes attack plans intended to compromise the system's resources.

AWDRAT (Architectural-Differencing, Wrappers,
Diagnosis, Recovery, Adaptivity and Trust-Management)
Howie Shrobe(MIT) and Bob Balzer(Teknowledge)



New Ideas

- **Model** intended application behavior, and use the models to:
 - **Block** harmful actions by using dynamic wrappers and architectural differencing.
 - **Diagnose** unintended behavior
 - **Recognize** and **Preserve** critical data

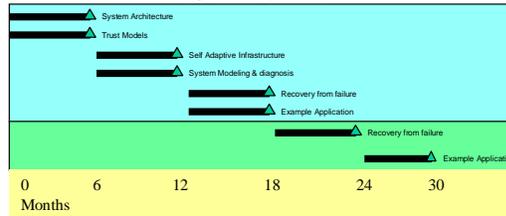
Impact

- Provides “cognitive immunity” to both intentional and accidental compromises
- Applications actively check that their behavior is consistent with their goals



Applications use a trust model to make rational choices about resource usage.

Schedule



2.4. Cortex

Honeywell: David Musliner

Cortex is based on the proposition that both success and security are defined by the computing mission, and that the computing mission changes over time. Therefore, only mission-aware automation can effectively manage critical systems under attack. Cortex will actively monitor and reconfigure a computing network to continuously optimize its performance of a computing mission. Cortex will use its awareness of its own resources, the changing character of the threat, and the changing demands of the mission, to maintain a dynamic balance between security and mission performance. Cortex will plan for contingencies; detect and classify threats as they occur; execute real-time responses to restore mission-critical services; and learn how to improve over time. The Cortex approach to reliable, self-regenerative systems will provide major benefits for mission-critical applications, including improved mission effectiveness, increased survivability, reduced resource margin requirements, and reduced overall cost. Cortex will provide these user benefits by combining three innovative capabilities:

- Scalable Coherent State Estimation: Cortex will use highly scalable qualitative probabilistic algorithms to combine the noisy, uncertain outputs from numerous system

sensors into an accurate and coherent estimate of system state. Using its built-in model of the system's computing mission, Cortex will diagnose the root cause of system problems and assess their potential mission impact, to guide tailored, optimized responses.

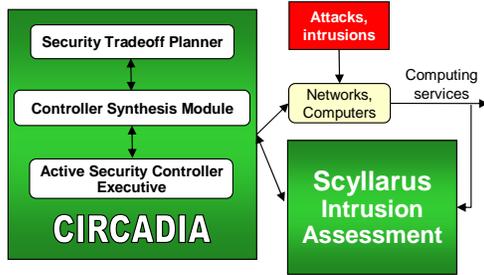
- Mission-Optimized Planning and Response: Cortex will use cognitively-inspired mission-aware planning algorithms to derive proactive response plans that optimize the system's mission performance during disruptions and attacks. Cortex will respond to 100% of the diagnosed attacks and faults. Furthermore, the planning algorithms will run online to continuously improve the system's resource allocation and response strategies, providing both self-regenerative and self-optimizing behaviors in a unified planning framework.

- On-line Learning: Cortex will begin operations with models of the computing system it controls, its mission, and the faults and attacks that may disturb it. Over time, Cortex will use statistical and structural learning algorithms to continually refine those models, improving the accuracy of its self-awareness and mission-awareness. Cortex will also use active probing to proactively self-test, self-diagnose, and automatically protect against potential faults and vulnerabilities, continually improving its robustness and security through self-discovery.

The Cortex concept builds on Honeywell's extensive prior experience in cyber security, fault tolerant systems, and cognitive systems, including Scyllarus and Circadia. With significant technical advances in mission-awareness, scalability, and continuous learning, Cortex will incorporate all of our experience into a unified self-regenerative system that will exceed DARPA's success criteria for cognitive immunity and self-healing.

Cortex will be demonstrated in increasingly complex scenarios at nine-month intervals. The demonstrations will be performed in Honeywell's Cyber Security Lab facility, using networks of computers performing simulated mission-critical computing tasks under a variety of attack, failure, response, and recovery scenarios.

CORTEX – Mission-Aware Closed-Loop Cyber Assessment and Response



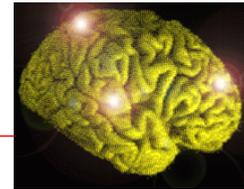
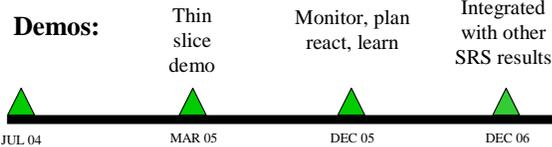
NEW IDEAS

- System Reference Model drives intrusion assessment, diagnosis, and response.
- Automatically search for response policies that optimize tradeoff of security against mission ops.
- “Taste-tester” server redundancy supports robustness and learning from new attacks.

IMPACT

- High confidence intrusion assessment and diagnosis.
- Pre-planned automatic responses to contain and recover from faults and attacks.
- Automatic tradeoffs of security vs. service level & accessibility.
- Learns to recognize and defeat novel attacks.

SCHEDULE



Honeywell Laboratories

3. Granular, Scalable Redundancy

Self-regenerative systems must operate even when damaged. Today, the keystone technique for doing this is to maintain multiple copies of selected system components and, if an attack damages some, to dynamically switch to using other, undamaged components. This technique imposes stringent coordination requirements on system components: these requirements may degrade performance to unacceptable levels. In this technical area, the program aims to develop new techniques that allow the necessary coordination to occur but with levels of performance that are needed by high-performance military systems.

3.1. Scalability, Accountability and Instant Information Access for Network-Centric Warfare

Johns Hopkins University: Dr. Yair Amir

Purdue University Subcontract: Dr. Cristina Nita-Rotaru

Network-centric warfare calls for survivable command control communication and intelligence (C3I) systems that are resilient to a broad range of attacks. The focus of this project is to construct a realistic solution for the broad malicious attack problem where part of the C3I system is compromised.

The project targets three main limitations with current solutions: they are not scalable to high latency wide area networks underlying C3I systems; they have no protection against malicious clients providing incorrect input that is within their authority; and they often

unnecessarily delay applying updates, withholding important information from clients until updates can be globally ordered.

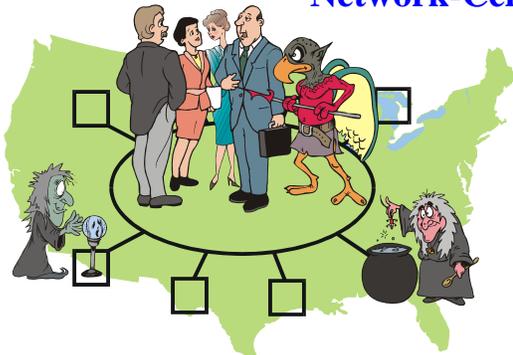
From a research perspective, there is a broad class of distributed data management applications based on replication infrastructure. This project takes the C3I problem as a representative example of this broader class.

The key innovations of our approach include:

- *Scalable wide-area intrusion-tolerant architecture*: By inventing a hierarchical approach in which Byzantine replication is used locally in each site, and efficient fault tolerant replication is used on the wide area network, we overcome the strong connectivity requirements and multiple all-peer exchanges of current Byzantine replication solutions. Symmetric Byzantine replication in conjunction with threshold cryptography is used in each site to create one logical trusted entity, over which the non-malicious tolerant replication can be safely used. The effects of malicious server replicas are then confined to the local site.
- *Accountability for updates*: Once bad data is discovered, we identify the client that injected it and quickly mark corrupted and suspected data. We can then backtrack and regenerate the C3I state based on non-corrupted and/or non-suspected data, and identify the extent of potential damage. Accountability for updates also provides protection against a complete site compromise, enabling a reduction in the number of replicas for a slightly higher risk and better performance.
- *Instant Information Access*: Our architecture propagates updates to other sites as soon as network connectivity exists and exploits commutative update semantics to efficiently make update effects available immediately. In contrast, Byzantine replication solutions may only provide access to the effects of updates that are globally ordered on the wide area network.

The resulting system will have considerably better performance and much higher availability than existing symmetric solutions and offer a clear path for technology transition.

Scalability, Accountability and Instant Information Access for Network-Centric Warfare



New ideas

First scalable wide-area intrusion-tolerant replication architecture.

Providing accountability for authorized but malicious client updates.

Exploiting update semantics to provide instant and consistent information access.

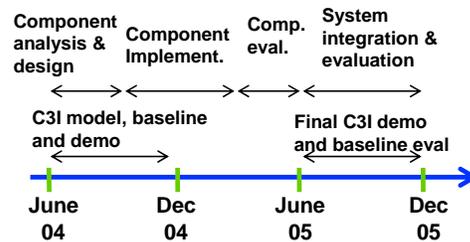
Impact

Resulting systems with at least 3 times higher throughput, lower latency and high availability for updates over wide area networks.

Clear path for technology transitions into Military C3I systems such as the Army Future Combat System.

Johns Hopkins University & Purdue University

Schedule



<http://www.cnds.jhu.edu/funding/srs/>

3.2. Increasing Intrusion Tolerance via Scalable Redundancy

CMU: M. K. Reiter G. R. Ganger P. Narasimhan A. Ailamaki C. Cranor

We propose to dramatically increase the scalability of fault- and intrusion-tolerant services, particularly in the efficiency of tolerating significant numbers of failures and compromises. Specifically, we will push the state-of-the-art in at least the following ways.

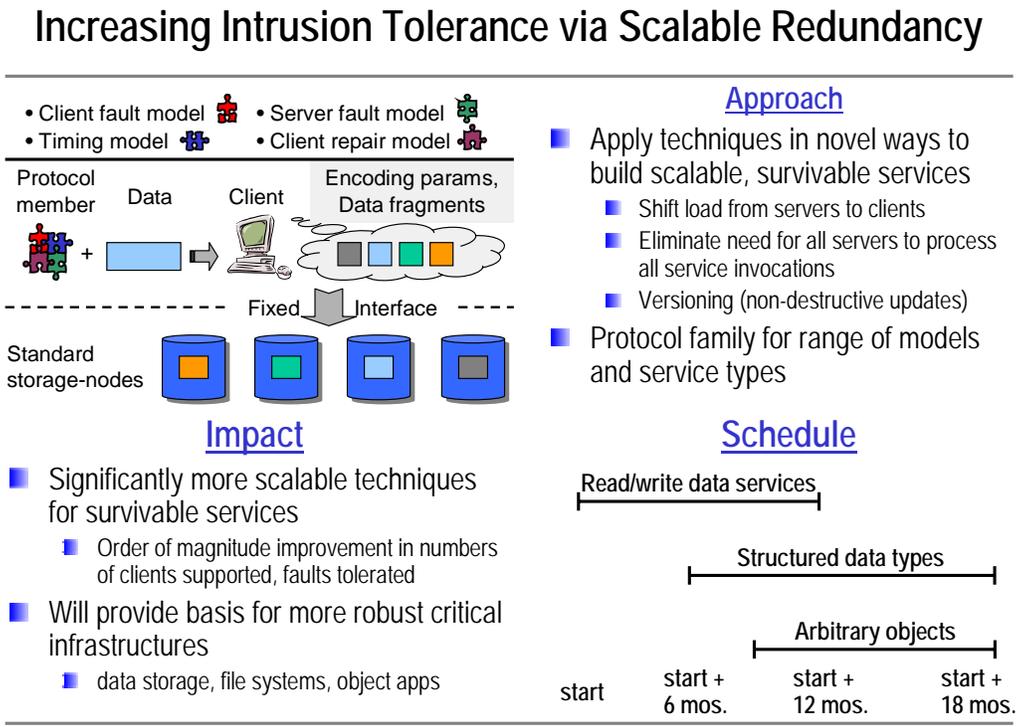
1. We will develop novel protocols for distributed data storage that offer linearizable data access, wait-free liveness, tolerance of Byzantine client and server failures (corruptions), and far superior latency and throughput as the system scales. Relative to the state-of-the-art, our protocols will yield at least a threefold improvement in access latency, even for relatively small numbers of servers, and this gap will grow as the number of servers increase. Simultaneously, they will at least double the throughput achieved by the current state-of-the-art, again even for relatively small server configurations, and will scale better as the system grows.

2. We will develop protocols for highly-resilient distributed object-based systems that offer linearizable method invocations, including nested invocations in which one object is invoked by another, where some replicas of each may be corrupt. The result will be unprecedented support for general intrusion-tolerant service construction. Our protocols will strive for sublinear growth in access cost as the number of object replicas grows, as compared to the linear-or-worse access costs offered by existing approaches.

3. We will develop specialized protocols for important object types (e.g., directories and indices) that improve upon the scalability of general object mechanisms by exploiting the best features of both protocol families described above. Such specialization reduces costs to those required for the needed functionality.

This project seeks to conquer a fundamental limitation of prior techniques: the need to synchronize all servers' states, which severely restricts a service's ability to scale. While the need to coordinate server state is to some extent unavoidable, in order to ensure consistent service semantics, we will significantly improve on the state-of-the-art via novel applications of techniques from other domains: shifting load from servers to clients; eliminating the need for all servers to process all service invocations; and using versioning (non-destructive updates) at servers to avoid the need to order updates proactively.

The breadth of services types we consider—ranging from read-write data storage to arbitrary object computations—allows our results to be applied to implementations specialized to the needs of a particular system. This breadth also introduces challenges on several fronts. For example, the distinction between read-write objects and more complex ones fundamentally changes the consistency and liveness properties that can be achieved when implementing them, and the associated performance costs. As another example, an object-oriented approach introduces the possibility of one object being invoked from another, which magnifies a corrupt object replica into a corrupt client for another object.



Carnegie Mellon: M. K. Reiter G. R. Ganger P. Narasimhan A. Ailamaki C. Cranor

3.3. Quicksilver

Cornell: Profs K. Birman, P. Francis and J. Gehrke

Raytheon: Dr. L. DiPalma and P. Work

A wave of “network centric warfare” (NCW) applications are in the planning stages. Unfortunately, off-the-shelf commercial products are difficult to deploy when scaled to very large environments, costly to administer, perform poorly under stress, and lack capabilities required of applications that may need to withstand problems ranging from mundane mishaps to attacks. Our effort brings together a unique alliance to build a new generation of technologies to bridge the gap. The QuickSilver will support adaptive, self-repairing, self-managed applications where scalability and robustness are central requirements.

This work targets two categories of applications: network centric warfare applications, and lightweight sensor systems, in which small devices are scattered in a theatre of operations and must organize themselves to provide useful data to military strategists and commanders. But the needs seen in these settings also occur in many other government and commercial systems. Progress will lead to improvements of the commercial technology base and will benefit many kinds of users.

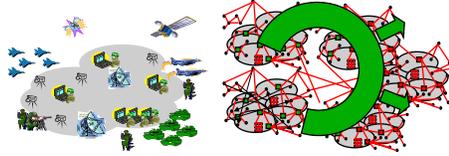
QuickSilver will work within emerging standards to the greatest degree possible. The development effort is focused on extending the Web Services architecture with functionality missing in today’s products. This approach will let developers leverage the power of broadly accepted commercial solutions while also gaining the assurance properties required in demanding settings.

Our effort is informed by years of close collaboration and dialog between Cornell researchers and the Joint Battlespace Infosphere (JBI) team at Air Force Research Laboratory in Rome New York. Raytheon team members point to decades of activity directed towards a wide range of military problems, with a current emphasis on issues seen in Navy Surface and Underwater Sensors and systems.

Our group has a strong track record of technology transfer. For example, work Cornell did as part of the Hiper-D program back in 1989 came to play a critical role in the onboard systems for the AEGIS warship, and also transitioned into the New York and Swiss Stock Exchanges, the French Air Traffic Control system, and other many projects. Today, we maintain a close dialog with companies like Microsoft and IBM, and with demanding application developers at companies like Amazon.com. This rich network of connections should keep QuickSilver focused on the right problems and help with technology transition when we reach that stage.

QuickSilver: Middleware for Scalable Self-Regenerative Systems

Innovative Technologies for Building Adaptive, Self-Repairing Global Information Grid Applications

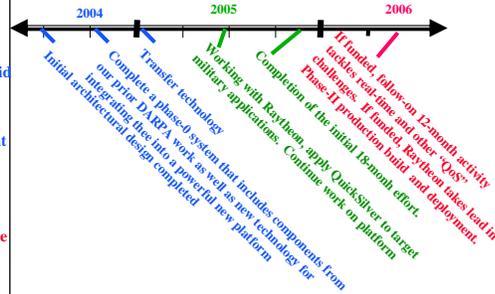


Impact

- * QuickSilver will enable rapid progress on global information grid platforms and applications, such as the Air Force JBI.
- * The system brings a new probabilistic style of state capture and diagnosis to bear on the challenges of deployment and management of large-scale applications
- * A Raytheon-led component effort is focused on dialog with real users in the Air Force and Navy
- * The technology base will enable dramatic progress in some of the most demanding military settings we face, while also benefitting civilian developers of critical infrastructure applications

QuickSilver Ideas

- * Augment Web Services architecture with missing technology components to support robust, scalable GIG applications
- * QuickSilver uses probabilistic techniques to dynamically sense distributed system state, automate configuration and adaptation under stress, and repair damage after a failure or disruption.
- * Incorporates “best of breed” solutions from prior DARPA work. Epidemic gossip protocols are robust against denial of service attacks that cripple conventional networks and applications.
- * Potential for revolutionary advances in tools for building and managing large, complex distributed systems, and for securing them



Cornell University: K. Birman (PI), P. Francis, J. Gehrke, R. VanRensse, W. Vogels. Raytheon: L. DiPalma, P. Work

4. Reasoning about the Insider Threat

Even if a military information system functions perfectly, a malicious operator is “inside” the system and can subvert a mission by giving the wrong command at a critical juncture. In this technical area, the program will develop technology allowing a system to estimate the likelihood that a military system operator will become malicious. While it is probably not possible to achieve 100% accuracy in such an estimate, progress in this area is critical to safeguarding deployed military systems, and can also serve as a deterrent to military system operators that might consider an inside attack.

4.1. Detecting and Preventing Misuse of Privilege

Teknowledge: Bob Balzer

MIT: Howie Shrobe

Project Description

The project assumes that the insider has all the access, privileges, and knowledge needed for an attack and focuses on detecting the malicious behavior required to mount that attack. This detection will be based on a unique set of sensors that monitor military user actions and an advanced malicious behavior detector that analyzes the military user history relative to a role-based model of expected behavior. This model will identify both the types of behavior expected in a situation and the means for assessing the appropriateness of the particular behavior observed. The assessment will use a wide variety of mechanisms for determining the appropriateness of an action such as safety models, “plant” models, design rules, best practices, and heuristics. This analyzer will detect both intentional and accidental actions that harm the system. A suspicious behavior

detector will differentiate the two by inferring user goals from the behavior and identifying the set of plans consistent with that behavior.

Two unique capabilities result from detecting attacks based on model-based predicted harm (about to be) caused to a system:

1. There is no need to update the defense as new insider attacks are discovered or new ways to obfuscate them are invented
2. Attacks based on corrupted operand values or the situation in which operations are invoked can be detected and blocked.

Deliverables

Misuse Detection Architecture: a generic architecture for monitoring operator behavior in military legacy systems at the level of application-specific commands or directives invoked by the operator, for matching that behavior against role-based plans, for modeling the effect of those commands or directives on the state of the legacy system, for assessing the benefit or harm of those effects, and for matching those effects and assessments against a set of insider attacks.

Operator Behavior Monitor: a component that mediates the communication between a legacy system's GUI and the system itself to extract the application level commands or directives initiated by the user/operator through that GUI so that they can be screened for harmful effects before being processed by the legacy system.

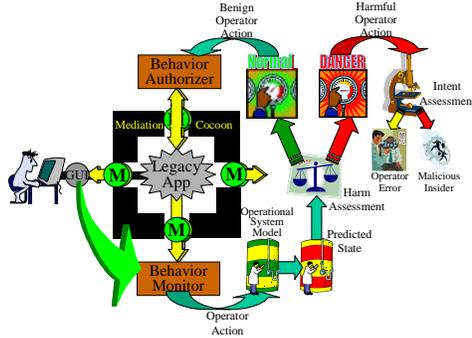
Matching Operator Behavior against Role-Based Plans: a component that compares operator behavior traces to behavior traces from operator and attack plans.

Operational System Model: an operational system model for a legacy application - initially constructed from propositional rules - from which both the predicted state of the system, and the likelihood of harm resulting from the change of state can be predicted.

Malicious Behavior Detector: a suspicious behavior detector that differentiates between accidents and malicious behavior by inferring user goals from the observed harmful behavior, recent historical behaviors, and the set of plans consistent with the larger behavior context.

Detecting and Preventing Misuse of Privilege

Bob Balzer(Teknowledge) and Howie Shrobe(MIT)



Impact

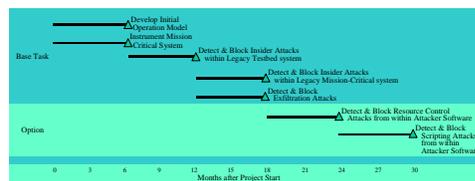
- Harmful actions blocked before damage inflicted
- Defenses don't need to be updated for new (zero-day) insider attacks or obfuscation techniques
- Insider attacks can be differentiated from operator error

TEKNOLEDGE

New Ideas

- **Monitor** Military User/Operator Actions
- **Predict** effect of user/operator action by applying it to operational system model
- **Block** harmful actions
- Assess intent of harmful actions to **differentiate** operator error from malicious insider

Schedule



4.2. Mitigating the Insider Threat using High-dimensional Search and Modeling

Telcordia: Eric Van Den Berg

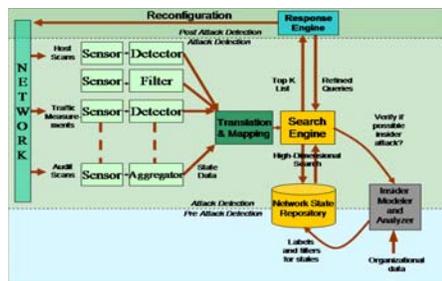
Rutgers: Raj Rajagopalan

Insider attacks cause over 70% of today's security breaches and often go undetected for long periods of time. Existing security technologies such as firewalls or Intrusion Detection Systems (IDS) do not provide adequate defenses against insider threats because they are oriented towards attacks originating from outside the enterprise. Unfortunately, insider attacks may begin from any of numerous potential attack points in the enterprise and have too many parameters to be monitored that existing systems cannot handle. Some existing technologies depend on an exact match of a known attack pattern (a "signature") and cannot detect even slight variations of those attacks. Others can detect anomalies in monitored statistics but cannot deal with a large number of attributes due to the problem of high dimensionality. We propose to build a system that can efficiently detect anomalies in high dimensional state spaces and that, by inferring attacker goals and targets, synthesizes appropriate pro-active responses to protect (correct spacing)critical services while minimizing collateral damage. We propose to achieve this goal with a series of steps and components outlined below:

- Collect data from numerous diverse sensors monitoring various layers of the military information system from physical to network layer to application layer and end-host sensors, making it very hard for any suspicious insider behavior to avoid triggering some sensor alerts. We will deliver a design document and prototype software for the sensors.

- A **network history repository**, which contains historical states of the system that are annotated with attack and precursor information. We will deliver a design document and prototype of a network state description language in terms of collected sensor data.
- A **high-dimensional search engine** operating on the network history repository, that is based on dimension reduction techniques such as Singular Value Decomposition (SVD). Given the current state of a large network, the search engine will create a ranked “Top K” list of annotated states that are all “similar” to this state. The search engine detects insider threats by creating clusters of “similar” states in a high dimensional space, which represent previously seen threat states or their precursors. Extrapolating from our experience, this engine will be able to detect, within minutes, over 40% of insider threats in large networks. We will deliver a design document and proof-of-concept prototype software for the search engine.
- A graph-based **insider threat modeling and analysis** tool, which identifies potential insider attack points and attack scenarios in a network by modeling the effort required to acquire knowledge of internal details, and provides the required “experience” for the pre-attack training phase. This helps reduce noise in data and prune the size and number of clusters. SUNY Buffalo will deliver an initial prototype for this tool.
- A **response engine**, which performs an impact analysis of the potential attack on critical services and automatically synthesizes a response that minimizes collateral damage, thwarting the predicted attack within minutes. We will deliver a design document and proof-of-concept prototype of the response engine. If the project is successful, we expect to be able to transition the research results and (close up spacing)technology to interested DoD customers. An additional side effect of the success of this project may be a breakthrough in the ongoing war against multi-dimensional virus-borne attacks.

Mitigating the Insider Threat using High-dimensional Search and Modeling



New Ideas

- Network State description based on large sensor network, monitoring services and devices at multiple layers
- Search Engine approach for detection of anomalies and known (insider) attacks, which overcomes dimensionality problem limiting current methods
- Static identification of potential attack points and insider attack scenarios in a network, using Insider Modeler and Analyzer (SUNY Buffalo)
- Analyze impact of detected attack, leveraging logical Response Engine to synthesize possible reconfigurations and their side-effects

Impact

- New high dimensional Anomaly Detection System, which scales to large, internet size networks
- Extrapolating from our experience, we expect the system to detect over 40% of insider threats in large networks
- Thwart insider attacks quickly, leveraging logical Response Engine, to protect critical services and minimize side-effects
- Transfer the research results to interested customers for immediate use once the approach has been validated

Schedule & Deliverables

- Q1, Q2: Design sensors, Network State description, Search Engine and Response Engine; Deliverable: Design document
 - Q3, Q4: Develop prototype Sensor Network, Translator, Search Engine, Response Engine and Insider Modeler and Analyzer; Deliverable: Prototype software
 - Q5, Q6: Experimentation: Test-bed construction, experiment design, experiment execution. Deliverables: Methodology document, Experiment Report and Demonstration
 - Final deliverable: Final Report
- We will further provide demonstrations where possible, and any publications resulting from this research