DDDAS-based Resilient Cyberspace

PIs: Salim Hariri, Youssif Al-Nashif, Ricardo Valerdi
University of Arizona
Stacy Prowell, Oak Ridge National Laboratory

Collaborator:
Erik Blasch, Air Force Research Laboratory

This project is partially supported by
AFOSR DDDAS Award Number FA95550-12-1-0241
Presentation Outline

• Motivations and Objectives
• Resilient DDDAS (rDDDAS) Architecture
  – Experimental Results and Evaluation
• Resilience Analysis and Quantification
• rDDDAS Applications
  – Resilient Crisis Management
  – Resilient Smart Grid (rSG)
• Conclusions and Future Work
Motivation - Resilience

• Human endpoint devices are the most vulnerable devices and the easy one to penetrate and exploit.

• Software, hardware, websites, cloud services & DDDAS all will have errors, vulnerabilities that can be exploited.
  – DDDAS paradigm provides the ability to continuously monitor, analyze, diagnose and respond in a timely manner to provide resilient cyberspace operations

• We expect DDDAS paradigm to deliver automated and proactive management for a wide range of mission critical applications, crisis management applications, and scientific and engineering applications
  – Protecting the operations of DDDAS applications will be even more critical
Relevant to Air Force Technical Horizons

- The project supports three Potential Capability Areas (PCAs) identified in the Air Force Technical Horizons report:
  - PCA1: Inherently Intrusion-Resilient Cyber Systems
  - PCA2: Automated Cyber Vulnerability Assessments and Reactions
  - PCA3: Decision-Quality Prediction of Behavior
rDDDAS: DDDAS-based Resilient Cyberspace

Closed loop architecture and Continuous feedback
Moving Target Defense Strategies

• Address Space Randomization
• Instruction Set Randomization
• Data Randomization
• Execution Environment Randomization
  – Change Programming Language
  – Change OS and Middleware
  – Change Resources
Software Behavior Encryption (SBE)

• Diversity
  – Hot Shuffling software variants at runtime
  – Variants are functionally equivalent, behaviorally different

• Redundancy
  – Multiple replicas on different physical hardware

• Random Selection and Shuffling of Variants
ATTACK Window for SBE Algorithm

Successful Attack
- Probing
- Constructing
- Launching

Only one Version

Thwarted Attack
- Probing
- Constructing
- Launching

Version 2
Version 4
Version 1
R-DDDAS TEST-BED AND RESULTS
rDDDAS Environment

Resilient Cloud Middleware

- Diversity Level
- Redundancy Level
- Shuffling Rate

Configuration Engine

Applications/Resources

- Application Repository
- VM Images Repository

Application Supervisor

- Observer
- Analyzer

Application Execution Environment 1

- VM1 (V1)  VM1 (V1)  VM1 (V1)

Application Execution Environment 2

- VM1 (V1)  VM1 (V1)  VM1 (V1)

Application Execution Environment n

- VM1 (V1)  VM1 (V1)  VM1 (V1)
Phase 1 Map

Supervisor 1
- Master 1
  - Worker 1 [V4]
  - Worker 2 [V7]
  - Worker 3 [V2]

Supervisor 2
- Master 2
  - Worker 4 [V9]
  - Worker 5 [V1]
  - Worker 6 [V6]

Supervisor 3
- Master 3
  - Worker 7 [V3]
  - Worker 8 [V5]
  - Worker 9 [V8]

Data store for VM images

Invoking Virtual Machines

Check Pointing

Supervisor Selection
Validation Results

- Developed an experimental rDDDAS environment
- MapReduce Application
- Linear Equation Solver Application
- Muibench
Application 1 – MapReduce (MR)

- Large-Scale Data Processing
- MapReduce provides
  - Automatic parallelization & distribution
- MapReduce Wordcount program

<table>
<thead>
<tr>
<th>Slave Operating System</th>
<th>Physical machine 1</th>
<th>Physical Machine 2</th>
<th>Physical Machine 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>Linux</td>
<td>Linux</td>
<td>Linux</td>
</tr>
<tr>
<td>Windows</td>
<td>Windows</td>
<td>Windows</td>
<td>Windows</td>
</tr>
<tr>
<td>Programming language</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java</td>
<td>V1</td>
<td>V5</td>
<td>V9</td>
</tr>
<tr>
<td>C++</td>
<td>V2</td>
<td>V6</td>
<td>V10</td>
</tr>
<tr>
<td></td>
<td>V3</td>
<td>V7</td>
<td>V11</td>
</tr>
<tr>
<td></td>
<td>V4</td>
<td>V8</td>
<td>V12</td>
</tr>
</tbody>
</table>

SBE example - MapReduce

Machine 1

Phase 1
1st Map Job

V3
Acceptance Test
TA1

Phase 2
2nd Map Job

V1
Acceptance Test
TA2

Phase 3
Final Map/Reduce

V1
Acceptance Test
TA3

Machine 2

Phase 1

V7
Acceptance Test
TA1

Machine 3

Phase 1

V2
Acceptance Test
TA1

Checkpoint-select the best output for next step

Final Output

Checkpoint-select the best output for the final output

AFOSR DDDAS PI Meeting, December 1-3, 2014
MapReduce – Attack Scenarios

During validation, SM checks current environment and if okay, DSSC starts the application execution cycle.

Case 1: During validation, SM detects an error in V4 and DSSC selects the first error free output from v5 or v12.

Case 2: During validation, SM detects compromised results of V9 and DSSC selects the first error free result from V3 or V7.

Case 1: Resilience against Dos Attacks

Denial of Service attack on Windows VM-6

<table>
<thead>
<tr>
<th></th>
<th>Response Time (in seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Without DoS attack</td>
</tr>
<tr>
<td>Without MTD</td>
<td>95</td>
</tr>
<tr>
<td>With MTD</td>
<td>105</td>
</tr>
</tbody>
</table>

Case 2: Resilience against Insider Attacks

Compromise attack on Linux VM-1

<table>
<thead>
<tr>
<th></th>
<th>Response Time (in seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Without Insider attack</td>
</tr>
<tr>
<td>Without MTD</td>
<td>95</td>
</tr>
<tr>
<td>With MTD</td>
<td>105</td>
</tr>
<tr>
<td>% increase in response</td>
<td></td>
</tr>
<tr>
<td>time with MTD</td>
<td></td>
</tr>
</tbody>
</table>

Application 2: Linear Equation Solver

- Supervisor randomly selects a phase timer.
- All three nodes run the three phases independently in independent versions.
- At the end of each phase, checkpoints are passed to the supervisor.
- Supervisor selects the most recent and correct checkpoint and passes to the SBE machine.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Physical Node 1</th>
<th>Physical Node 2</th>
<th>Physical Node 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>V1</td>
<td>V7</td>
<td>V10</td>
</tr>
<tr>
<td>Linux</td>
<td>V4</td>
<td>V8</td>
<td>V11</td>
</tr>
<tr>
<td>Windows</td>
<td>V2</td>
<td>V12</td>
<td>V14</td>
</tr>
<tr>
<td>Linux</td>
<td>V5</td>
<td>V9</td>
<td>V15</td>
</tr>
<tr>
<td>C</td>
<td>V3</td>
<td>V6</td>
<td>V16</td>
</tr>
<tr>
<td>C++</td>
<td>V2</td>
<td>V8</td>
<td>V11</td>
</tr>
<tr>
<td>Fortran</td>
<td>V7</td>
<td>V10</td>
<td>V14</td>
</tr>
<tr>
<td></td>
<td>V13</td>
<td>V12</td>
<td>V17</td>
</tr>
<tr>
<td></td>
<td>V16</td>
<td>V15</td>
<td>V18</td>
</tr>
</tbody>
</table>
Application 2 - Overhead

Table I- Overhead in Application 2

<table>
<thead>
<tr>
<th>Execution Time in seconds without SBO</th>
<th>Time</th>
<th>OH</th>
<th>Time</th>
<th>OH</th>
<th>Time</th>
<th>OH</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>218</td>
<td>9%</td>
<td>248</td>
<td>24%</td>
<td>276</td>
<td>38%</td>
</tr>
<tr>
<td>800</td>
<td>838</td>
<td>5%</td>
<td>890</td>
<td>11%</td>
<td>988</td>
<td>24%</td>
</tr>
<tr>
<td>1500</td>
<td>1568</td>
<td>5%</td>
<td>1624</td>
<td>8%</td>
<td>1663</td>
<td>11%</td>
</tr>
<tr>
<td>3600</td>
<td>3671</td>
<td>2%</td>
<td>3847</td>
<td>7%</td>
<td>3890</td>
<td>8%</td>
</tr>
</tbody>
</table>

SBE Execution Flow

Phase 1

Node 1

V 2

Acceptance Test

Node 2

V 3

Checkpoint Passing

Node 3

V 6

Phase 2

Node 1

V 1

Acceptance Test

Node 2

V 4

Checkpoint Passing

Node 3

V 5

Phase 3

Node 1

V 2

Acceptance Test

Node 2

V 3

Checkpoint Passing

Node 3

V 6

Normal Operation

Replica Compromised e.g. on Node 1

All replicas compromised

Previous checkpoint selected
Application 2 – Attack Scenarios

- DoS attack on V1
- Insider attack on Supervisor 2
- Compromise of two Virtual Machines

*Illustration of Attack scenario 1*

The SBE algorithm is started and the above versions run normally in Phase 1. During Phase 2, a DoS attack is launched on V1. The acceptance test detects this and selects the output of V8 for the next stage. During Phase 3, a new set of versions is selected. None of them are compromised and the execution continues normally.
Application 3 – MiBench Benchmarks

- It consists of C programs from six categories each targeting a specific area
  - Basicmath (Automotive and Industrial Category)
  - Dijkstra’s algorithm (Network category)

<table>
<thead>
<tr>
<th>Physical Machine Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Linux</th>
<th>Windows</th>
<th>Linux</th>
<th>Windows</th>
<th>Linux</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version number</td>
<td>V1</td>
<td>V2</td>
<td>V3</td>
<td>V4</td>
<td>V5</td>
<td>V6</td>
</tr>
</tbody>
</table>
Basicmath - Overhead for SBE with three phases

![Graph showing the overhead for SBE with three phases over the number of iterations. The overhead decreases significantly as the number of iterations increases.]
Dijkstra’s Algorithm - Overhead for SBE with three phases

![Graph showing overhead percentage vs number of iterations for 2, 3, and 4 phases.]

Resilience Modeling and Analysis

- Resilience Quantification
- Trust Modeling and Evaluation
r-DDDAS Design Challenges

• We expect DDDAS to be adopted to implement Mission Critical Applications, Command and Control Applications in industry (e.g., IBM Smart City or Environment), government (e.g., Crisis Management), etc.

• How do you determine the appropriate resilience requirement of such critical DDDAS applications?

• How do you trust, quantify trust of users, data, results, etc.? 
Resilience Cloud Analysis Framework

• STEP 1: Identifying the attack vectors for applications with respect to the system attack vectors
• Step 2: Resilience Quantification
System attack surface for windows

1. Open Sockets
2. Open RPC endpoints
3. Open named pipes
4. Services
5. Services running by default
6. Services running as system
7. Active web handlers
8. Active ISAPI filters
9. Dynamic web pages
10. Executable vdirs
11. Enabled accounts
12. Enabled accounts in admin group
13. Null sessions to pipes and shares
14. Guest account enabled
15. Weak ACLs in FS
16. Weak ACLs in Registry
17. Weak ACLs on share
18. VBScript enabled
19. JScript enabled
20. ActiveX enabled
System attack surface for Linux

1. open_TCP/UDP_socket
2. open_RPC_endpoint
3. service_running_as_root
4. service_running_as_non_root
5. setiud(setgid)_root_program
6. enabled_local_user_account
7. user_id=root_or_group_id=root_account
8. unpassworded_account
9. nobody_account
10. weak_file_permission
11. script_enabled
12. dynamic_webpage
13. symbolic_link
14. httpd_module
Linear Equations Attack Vectors

- Open sockets
- Weak file permissions
- Service-running as root
- RPC_endpoints
- Weak ACL’s on Registry (Only with windows)
STEP 2: Resilience Quantification

• Analytical Approach
• Discrete Event Simulation
Resilience Analysis

The system resilience $R$ is the ability of the system to continue providing its QoS as long as the impact of the attacks is below the minimum threshold $R$.

The impact $i_v(t)$ of a vulnerability $v$ is:

$$i_v(t) = \begin{cases} 0, & t < T_v \\ I_v, & t \geq T_v \end{cases}$$

Where $T_v$ is the time required for discovering the vulnerability and exploiting it, and $I_v$ is the impact of exploiting the vulnerability.
Resilience Analysis - Continue

The expected value of the impact of a vulnerability \( v \) is given by:

\[
E[i_v] = I_v \cdot Pr(A_v)
\]

Where \( A_v \) is the random variable that represent the occurrence of an attack exploiting vulnerability \( v \). We can go further:

\[
Pr(A_v) = Pr(A) \cdot Pr(U_v)
\]

\( A \) denotes the existence of an attacker who is trying to exploit the system, and \( U_v \) denotes the time needed to successfully exploiting the vulnerability \( v \).
Resilience Analysis - Continue

\[ i_{\text{system}} = E[i_{v1}] + E[i_{v2}] + \cdots + E[i_{vN}] \]

\[ = I_{v1} \cdot \Pr(A) \cdot \Pr(U_{v1}) + I_{v2} \cdot \Pr(A) \cdot \Pr(U_{v2}) + \cdots + I_{vN} \cdot \Pr(A) \cdot \Pr(U_{vN}) \]

\[ = \sum_{k=1}^{N} I_{v_k} \cdot \Pr(A) \cdot \Pr(U_{v_k}) = \Pr(A) \cdot \sum_{k=1}^{N} I_{v_k} \cdot \Pr(U_{v_k}) \]
Erik Blasch, Youssif Al-Nashif, Salim Hariri, Static versus Dynamic Data Information Fusion analysis using DDDAS for Cyber Trust, ICCS 2014.
Simulation Approach

SBE Version N = 3 versions with Denial of Service and Distribution Based ATK

Acceptance Testing
Trust Analysis and Quantification

• Trust can be defined as a state of mind consisting of
  – Expectancy of what the trustee will do
  – Believe expected behavior will be performed
  – Willingness to take risk in trustee

• Trust can be based on reputation
  – Collect from collaborating and interacting entities

• Trust a factor of technology and computations

\[ \text{End-to-end trust} = f (\text{Trust in infrastructure, Trust in entities}) \]

Which can be as simple as:

\[ \text{End-to-end trust} = (\text{Trust in infrastructure}) \times (\text{Trust in entities}) \]
Trust Stack

- Policies Enforcement
- Domain Trust Authority
- Collecting Raw Metrics
- Behavior Analysis Analysis (Situation Awareness)
- Authentication and Authorization
- Secure Communication

Erik Blasch, Youssif Al-Nashif, Salim Hariri, Static versus Dynamic Data Information Fusion analysis using DDDAS for Cyber Trust, ICCS 2014.
Domainless Trust Evaluation Architecture

Trust Authority

Mutual Authentication

ATM_A

Entity 1

ATM_B

Entity 2

End-to-End Communication
User Trust

• The initial user trust is a function based on the user rank and reputation at the time of adding it to the system.

\[ T(E, C) = T(User) \times T(Machine \text{ based on CVSS}) \]

- Trust measurement takes the value from 0 to 1
  - 0 represents the distrust and
  - 1 represents the blind or full trust.
- We adopt the NIST standard SP 800 four levels of trust:
  - Distrust: 0
  - Low trust: 0.33
  - Moderate Trust: 0.66
  - High Trust: 1
Adaptive Trust (continue)

Trust is evaluated by the Trust Authority

Self-Trust Evaluation:

\[ T_s(E, C) = T(\text{ATM}_E, C) \times \prod_{i=1}^{L} I_i(C) \times m_i(M_i) \]

Peer-Trust Evaluation:

\[ T_p(E, C) = \frac{1}{K} \sum_{j=1}^{K} \left[ T(\text{ATM}_j, C) \times T_{p,E_j}(E_j, C) \right] \]

C = Context (Mandatory Fields: Federal ID#; Optional Fields)
E = Evaluation Strategies / Enforcement (Rules – e.g.; Space, Time Policies)

- The values of the metric weight \( I_i \) for metric \( i \) is determined based on the feature selection technique, where:

\[ \prod_{i=1}^{L} I_i(C) = 1 \]
rDDDAS – Future Work

- r-DDDAS based Crisis Management
- r-DDDAS Smart Grid
- r-DDDAS based Hacker Web Monitoring and Analysis
- r-DDDAS based Hurricane Management System
r-DDDAS as a Cloud Service (rDaaS)

• How do you offer rDDDAS environments to implement a wide range of applications?
• How can you integrate design stage and runtime stage to achieve continuous integration of design and runtime?
rDDDAS-based Crisis Management

- Battle Management
- Nuclear Disaster Management
- Terrorist/Accident Management

- DDAS Analytics for Cybersecurity

Management Domain
- Decision Makers
- Domain Experts

Operations Domain
- Air Force
- Police
- Firemen
- First Medical Responders

Actions

Sensors Measurements
rDDDAS as a Service (rDaaS)

We are collaborating with:
• Youakim Badr, LIRIS Lab, INSA-Lyon, France
rDDDAS Components as a Service (rDaaS)

Application Requirement

Service Model

Risk Model

Annotation Model

Web Service Composition

Internal Changes

Security Policy

Context Model

External Changes

Business Domain

Security Objective

Context Model

generate

feedback

generate

affect

affect

affect

AFOSR DDDAS PI Meeting, December 1-3, 2014

AFOSR | Cloud and Autonomic Computing Center
rDDDAS-based Crisis Management
USE CASE SCENARIO – Crisis Management

rDaaS for Crisis Environment
Resilient Smart Grid (rSG)

Source: http://www.hitachi.com/environment/showcase/solution/energy/smartgrid.html
UA Smart Grid Testbed
UA Smart Grid Testbed

Resilient Smart Grid (rSG)
Resilient Approach

System Sensors

BACnet Control Station
DNP3 Control Station
Wi-Fi Control Station
ZigBee Control Station

Channel Selection

Feedback Communication

Output Controller

ZigBee
Wi-Fi
DNP3
BACnet

Outputs
Choreographer System Overview
sensor pool

resilient server

steam
vibration
radiation
pressure
temp

resilient communication unit

resilient management supervisor

resilient communication unit

resilient server

choreographer

configuration engine

scada system

historian server

data acquisition server

engineering workstation

filed devices pool

cooling
water condensing
signal processing
valves & motors

resilient management supervisor

reports

resilient sg
Hacker Web Monitoring and Analysis

• Develop autonomic monitoring and analysis of IRC hacker messages
• Identifying hackers relations, behaviors, and interactions.
• Log all IRC chats in Hadoop Big Data Analytics Environment.
• Identifying malicious IRC channels using Big data analytics
IRC Client based technique

IRC Network

IRC Network Topology
Features Extraction and Reduction from IRC Messages

IRC Messages

Text Extraction ➔ Language Detection

Meta Data Extraction ➔ Statistical Natural Language Parser ➔ Keyword Extraction ➔ Semantic Role Parser

Translate if not English ➔ Noise Reduction

Message Stream Processing:
- Mapping
- Aggregate
- Correlate
- Cluster
- Classify
- associate
A Sample Bot

* jabber_420 (jabber_420@Test-D7A3CF8.ece.arizona.edu) Quit (Client exited)
* jabber_420 (jabber_420@Test-D7A3CF8.ece.arizona.edu) has joined #test

<jabber_420> Hey Everyone !!
<kkkk> good morning ACL lab
<kkkk> Hello everyone
<kkkk> Hey Jabber
<kkkk> hey jabber
<jabber_420> hey
<kkkk> how are you doing ?
<jabber_420> I am doing good ! how about yourself ?

An **IRC bot** is a set of scripts or an independent program that connects to Internet Relay Chat as a client, and so appears to other IRC users as another user. An IRC bot differs from a regular client in that instead of providing interactive access to IRC for a human user, it performs automated functions.
# Key-word Dictionary

<table>
<thead>
<tr>
<th>0-day</th>
</tr>
</thead>
<tbody>
<tr>
<td>exploit</td>
</tr>
<tr>
<td>malware</td>
</tr>
<tr>
<td>trojan</td>
</tr>
<tr>
<td>virus</td>
</tr>
<tr>
<td>zeroday</td>
</tr>
<tr>
<td>zero-day</td>
</tr>
<tr>
<td>vulnerability</td>
</tr>
<tr>
<td>worm</td>
</tr>
</tbody>
</table>
### Channels

<table>
<thead>
<tr>
<th>Server</th>
<th>Port #</th>
<th>Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>irc.anonops.com</td>
<td>6667</td>
<td>#anonops</td>
</tr>
<tr>
<td>irc.evilzone.org</td>
<td>6667</td>
<td>#evilzone</td>
</tr>
<tr>
<td>irc.hack5.org</td>
<td>6667</td>
<td>#hak5</td>
</tr>
<tr>
<td>irc.undernet.org</td>
<td>6667</td>
<td>#cc-trade</td>
</tr>
<tr>
<td>irc.secco.org</td>
<td>6667</td>
<td>#hakology</td>
</tr>
<tr>
<td>irc.evilzone.org</td>
<td>6667</td>
<td>#securityoverride</td>
</tr>
<tr>
<td>irc.freenode.net</td>
<td>6667</td>
<td>#r_netsec</td>
</tr>
<tr>
<td>irc.freenode.net</td>
<td>6667</td>
<td>##security</td>
</tr>
<tr>
<td>irc.freenode.net</td>
<td>6667</td>
<td>#droidsec</td>
</tr>
<tr>
<td>irc.freenode.net</td>
<td>6667</td>
<td>#corelan</td>
</tr>
<tr>
<td>irc.efnet.org</td>
<td>6667</td>
<td>#security</td>
</tr>
<tr>
<td>cfyz6afpgfeirst.onion</td>
<td>6667</td>
<td>#agora</td>
</tr>
<tr>
<td>ixf6tm3pfbdv4n2b.onion</td>
<td>6667</td>
<td>#anonet</td>
</tr>
</tbody>
</table>
Number of Users / IRC Channel

number of users/ channel

channel 1
channel 2
channel 3
channel 4

number of users/ channel
Usage Vs Time of Day
User Activity Mapping
High/Low User Activity

High User Activity

Low users Activity
DCRA facilitates the Navy Tactical Cloud

- Simplifies content distribution management of different levels of access among pre-established groups
- Protects data at rest, even when devices / networks don’t have KGY devices
- Reduces bandwidth and Server requirements due to low “overhead” of SRK process

- Ashore
- Afloat
- Sensors & Platforms

DCRA--- Secure, Agile, Scalable, and Available

Graphic Source
Briefing, Charlie Suggs, PEO C4I, 01/14/2013
Big Data Analytics

Situation Awareness, Environ. Monitoring, Alerts, Threat Tracks, Plans

DB Software
Accumulo

Data Analytics Software
Visualization (OWF)  Storm  Hadoop  MapReduce

DCRA Middleware
Resiliency Middleware  SRK

Storage HW

Computation HW
Thank You
Questions?

H. Kurra, **Y. B. Al-Nashif**, Salim Hariri, “Resilient cloud data storage services”, 2013 ACM Int. Conference on Cloud and Autonomic Computing (CAC '13)


Erik Blasch, Youssif Al-Nashif , Salim Hariri,
Static versus Dynamic Data Information Fusion analysis using DDDAS for Cyber Trust, ICCS 2014.

Cihan Tunc, Farah Fargo, Youssif Al-Nashif, Salim Hariri, John Hughes, Autonomic Resilient Cloud Management (ARCM), 2014 ACM Int. Conference on Cloud and Autonomic Computing, (CAC '14)