What If We Got A “Do-Over?”
an Overview of CRASH and MRC

Howard Shrobe
Program Manager, DARPA I2O
We are divergent with the threat…

Public sources of malware averaged over 9,000 samples (collection of exploits, worms, botnets, viruses, DoS tools)
The US approach to cyber security is dominated by a strategy that layers security on to a uniform but architecture.

We do this to create tactical breathing space, but it is not convergent with an evolving threat.
Two Models of Survivability

Fortress (traditional)
- Impenetrable (hopefully)
- Monolithic
- Single Layer
- Rigid
- Immobile

Organism
- Many partial barriers
- Heterogeneous
- Defense in depth & Self Healing
- Adapts, Learns, Evolves
- Mobile

Approved for Public Release, Distribution Unlimited
Humans have Two Immune Systems: Innate and Adaptive

**Innate**
- Neutrophils
- Macrophages
- Dendritic cells
- Natural killer (NK) cells

**Adaptive**
- CD3/CD4 T cell
- CD3/CD8 T cell
- B cells
- CD19
- CD16/56

Fast, but inflexible, covers fixed sets of pathogen that are always present. Supports the adaptive immune system.

Slower, learns to recognize new sets of pathogens, distinguishes self from non-self, retains memory to guard against future attacks.

At least 20 - 30% of the body’s resources are involved in constant surveillance and containment.
## Biology and Computation: Two Design Styles

<table>
<thead>
<tr>
<th><strong>Computation</strong></th>
<th><strong>Biological</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Near Perfect Components</td>
<td>Fallible components</td>
</tr>
<tr>
<td>Core design formed in era of scarcity</td>
<td>Abundance of resources</td>
</tr>
<tr>
<td>Core design formed in isolated environment</td>
<td>Evolution in ecosystem of predators and parasites</td>
</tr>
<tr>
<td>Evolutionary pressure from market: price, performance and features</td>
<td>Evolutionary pressure from ecosystem: survivability</td>
</tr>
<tr>
<td>Self-regulation and adaptation rarely considered. Runs open-loop.</td>
<td>Self-regulation and adaptation are core mechanisms. Closed loop control.</td>
</tr>
<tr>
<td>No enterprise-wide survivability mechanisms</td>
<td>Diversity for population survival</td>
</tr>
<tr>
<td></td>
<td>Public-health systems in human society</td>
</tr>
</tbody>
</table>
Innate Immunity:
New hardware & operating system architectures
that eliminate all common technical vulnerabilities

Adaptive Immunity
Middleware that:
• Diagnoses root causes of vulnerabilities and builds situational assessment
• Quickly adapts & reconfigures
• Learns from previous attacks and gets better at self-protection

Population Diversity
Computational techniques that:
• Increase entropy in time and space
• Make every system unique
• Raise work factor of attacker for each system

Approved for Public Release, Distribution Unlimited
What If We Got A “Do-Over?”

Innate Immunity: Complete Mediation Through Hardware Enforcement

Approved for Public Release, Distribution Unlimited
Hardware Mediation

Conventional Computer

Memory

Register File

Result Data

Data

Operand 1

Operand 2

Tags Unit

Tag 1

Tag 2

Instruction

Trap Signal

Result Tag

Thread’s Principal

Access Rules

Bounds

Type

Provenance

Meta-computing system

Approved for Public Release, Distribution Unlimited
Tag Processing Unit is about 125K bits
125,000 SRAM bits < 16K Bytes

Note: L1 Dcache on Opteron is 64KB

Dual and Quad core Opterons
Separation via Compartments and Principals

- Each system component has its own **compartment** for its private data
- Each system component has its own **principal** for managing its private data
- System components may have “**satellite**” compartments and principals for controlled interactions with users and other system components
- Gates are used to control access to shared compartment and principals

---

Approved for Public Release, Distribution Unlimited
What If We Got A “Do-Over?”

Adaptive Immunity: Middleware for Diagnosis & Repair

Approved for Public Release, Distribution Unlimited
Adaptation through decision theoretic use of functional redundancy

Each method binds the settings of the parameters in a different way. Each task can be provided by several methods. Each method requires different resources. The resources used by the method have a cost.

User invokes a task. User's Utility Function. The specific binding of parameters has a value to the user. Service Quality Parameters. Each method binds the settings of the parameters in a different way.

Abstract task

Method<sub>1</sub> Method<sub>2</sub> Method<sub>n</sub>

Resource<sub>1,1</sub> Resource<sub>1,2</sub> Resource<sub>1,j</sub>

Resource Cost Function

The specific binding of parameters has a value to the user. The resources used by the method have a cost.

Expected Net benefit = Expected Benefit - Expected Cost

Probability of Success

Trust Model:
Risk of Failure
Likelihood of Success

Adaptation selects the method which maximizes expected net benefit.

* 

Approved for Public Release, Distribution Unlimited
DAIKON: Invariant Learning

“Failure Oblivious” Repair – Invariant Enforcement
E.g. Out of bounds write -> do nothing
Out of bounds read -> Return some array value

<table>
<thead>
<tr>
<th>ClearView Configuration</th>
<th>Page Load Time (seconds)</th>
<th>Overhead Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bare Firefox</td>
<td>7.5</td>
<td>1.0</td>
</tr>
<tr>
<td>Memory Firewall</td>
<td>11.0</td>
<td>1.5</td>
</tr>
<tr>
<td>Memory Firewall + Shadow Stack</td>
<td>14.9</td>
<td>2.0</td>
</tr>
<tr>
<td>Memory Firewall + Heap Guard</td>
<td>19.0</td>
<td>2.5</td>
</tr>
<tr>
<td>Memory Firewall + Heap Guard + Shadow Stack</td>
<td>22.7</td>
<td>3.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bugzilla Number</th>
<th>Presentations</th>
<th>Error Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>269095</td>
<td>6</td>
<td>Memory Management</td>
</tr>
<tr>
<td>285595*</td>
<td>4</td>
<td>Heap Buffer Overflow</td>
</tr>
<tr>
<td>290162</td>
<td>4</td>
<td>Unchecked JavaScript Type</td>
</tr>
<tr>
<td>295854</td>
<td>5</td>
<td>Unchecked JavaScript Type</td>
</tr>
<tr>
<td>296134</td>
<td>4</td>
<td>Stack Overflow</td>
</tr>
<tr>
<td>311710</td>
<td>12</td>
<td>Out-of-Bounds Array Access</td>
</tr>
<tr>
<td>312278</td>
<td>4</td>
<td>Memory Management</td>
</tr>
<tr>
<td>320182</td>
<td>6</td>
<td>Memory Management</td>
</tr>
<tr>
<td>325403*</td>
<td>4</td>
<td>Heap Buffer Overflow</td>
</tr>
</tbody>
</table>
### Evolutionary Programming for Patch Generation

 Automatically Finding Patches – Stephanie Forrest (CRASH)

<table>
<thead>
<tr>
<th>Program</th>
<th>Lines of Code</th>
<th>Path Length</th>
<th>Program Description</th>
<th>Vulnerability</th>
</tr>
</thead>
<tbody>
<tr>
<td>nullhttp</td>
<td>5575</td>
<td>768</td>
<td>webserver</td>
<td>remote heap overflow</td>
</tr>
<tr>
<td>openldap io.c</td>
<td>6519</td>
<td>25</td>
<td>directory protocol</td>
<td>non-overflow denial-of-service</td>
</tr>
<tr>
<td>lighttpd fastcgi.c</td>
<td>13984</td>
<td>136</td>
<td>webserver</td>
<td>remote heap overflow</td>
</tr>
<tr>
<td>atris</td>
<td>21553</td>
<td>34</td>
<td>graphical game</td>
<td>buffer overflow</td>
</tr>
<tr>
<td>php string.c</td>
<td>26044</td>
<td>52</td>
<td>scripting language</td>
<td>integer overflow</td>
</tr>
<tr>
<td>wu-ftp</td>
<td>35109</td>
<td>149</td>
<td>FTP server</td>
<td>format string</td>
</tr>
<tr>
<td>TOTAL</td>
<td>108784</td>
<td>1164</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Patched 55 out of 105 Bugs at an Average cost of $8 per patch
What If We Got A “Do-Over?”

Eliminating Monoculture
Monocultures are not survivable

- The attacker’s work factor is proportional to Entropy
  - When all systems are the same, a single attack disables them all
  - When a single system never changes, the same attack will work repeatedly

- We currently have a computational monoculture.
Dynamic Diversity makes a single host different from moment to moment

Address space randomization

Code and/or data blocks are periodically repositioned in memory so that attacker has to work harder to find a target. Garbage-Collected memory has the property inherently, new methods may optimize for increased entropy.

Instruction set randomization

There are multiple methods for achieving each goal ("n-version programming"). Each distinct method has different qualities of service. Method selection is driven both by preferences over QoS and by need for unpredictability.

Functional Redundancy & Decision theoretic dispatch

Code is encrypted as it enters memory and Decrypted as it enters the instruction cache (or translation buffer). Injected code in native instruction set is then encrypted and not executable. Encryption key can be varied by process and time.
What If We Got A “Do-Over?”

Moving beyond the host to the cloud
Cloud Computing Infrastructure

Your Software Lives Here

Virtual Machine

Multicore Chip

Core

Shared L3 Cache

Memory Controller

1U Blade in Blade Server

Modular Data Center

Blade Server Racks

Blade Server Network

Approved for Public Release, Distribution Unlimited
The drive to cloud computing:
- The White House released a 25-point plan in December for reforming government IT, and it included a **requirement that agencies adopt a cloud-first policy** for new IT deployments.
- “...we have found it is easier to secure the cloud... coming up with a defensible system is important and cloud computing is one way to help make that possible. By shifting to a cloud architecture, the US would save money and be better placed to protect vital computer networks.” (Gen Keith Alexander, Nov 7, 2011)

<table>
<thead>
<tr>
<th>Motivations</th>
<th>Concerns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Economic efficiency</td>
<td>Monoculture</td>
</tr>
<tr>
<td></td>
<td>Added attack surface of hypervisor</td>
</tr>
<tr>
<td>Manageability of large scale data centers</td>
<td>Concentration of hosts on high speed network with few internal barriers</td>
</tr>
<tr>
<td>Availability of “fungible computation” on demand. Use on demand vs. build to worst case.</td>
<td>“Fate sharing” among unrelated computations.</td>
</tr>
<tr>
<td>Centralization of data for common analytics</td>
<td>Obvious target for adversary &amp; single point of failure</td>
</tr>
</tbody>
</table>
Mission-oriented Resilient Clouds: Turning Liabilities into Assets

**TODAY**

Each host Acts in isolation, making the enterprise weaker than the sum of its parts

- “Box” oriented
- Vulnerable components
- Static
- Shared vulnerabilities
- Implicit trust within enclave amplifies vulnerabilities

**RESILIENT CLOUDS**

Acting as a community makes the enterprise stronger than the sum of its parts

- Components from CRASH program
- Moving target
- Resilience through diversity
- Collective defense & diagnosis across enclave dampens vulnerabilities
**Objective:** Sustain mission effectiveness. Different mission components have different security and performance needs and will make different trade-offs at run-time.

**Target:** Achieve at least 80% of optimal mission effectiveness while using at most 150% of the “base” resource requirements.
Resilient Clouds In Action

Initiating the Mission: Tasks are assigned to hosts and the network is configured to maximize mission effectiveness.

Shared Situational Awareness
Mission-aware Resource Optimization

Taskable Diversity      Mission-Aware Networking

Distributed, Highly Resilient Cloud Defense System

Approved for Public Release, Distribution Unlimited
Resilient Clouds In Action

Quorum Algorithm Over Multiple Instances of Task 13 Detects Violation

Shared Situational Awareness
Mission-aware Resource Optimization

Taskable Diversity          Mission-Aware Networking

Distributed, Highly Resilient
Cloud Defense System

Approved for Public Release, Distribution Unlimited
Migrate Task 16 to unaffected host

Resilient Clouds In Action

Shared Situational Awareness
Mission-aware Resource Optimization

Taskable Diversity
Mission-Aware Networking

Distributed, Highly Resilient
Cloud Defense System

Approved for Public Release, Distribution Unlimited
Increase communication network priority for host receiving task 16

Resilient Clouds In Action

Shared Situational Awareness
Mission-aware Resource Optimization

Taskable Diversity
Mission-Aware Networking

Distributed, Highly Resilient
Cloud Defense System

Approved for Public Release, Distribution Unlimited
Resilient Clouds In Action

All machines running Task 13 now suspect and directed to monitor Task 13

Host 1

Host 2

Host 3

Host 4

Host 5

Host 6

Host 7

Host 8

Tasks 3-5,16

Tasks 10-12

Tasks 1-3

Tasks 13-15

Tasks 6-8

Tasks 11-13

Tasks 1-5

Shared Situational Awareness
Mission-aware Resource Optimization

Taskable Diversity
Mission-Aware Networking

Distributed, Highly Resilient
Cloud Defense System

Approved for Public Release, Distribution Unlimited
Another Task 13 failure plus activity data from sandboxed Task 13 leads to more complete diagnosis & workaround for the problem.
Detectors, Patches & Workarounds for Task 13 vulnerability are distributed to all affected hosts.

Resilient Clouds in Action

Shared Situational Awareness
Mission-aware Resource Optimization

Taskable Diversity  Mission-Aware Networking

Distributed, Highly Resilient Cloud Defense System

Approved for Public Release, Distribution Unlimited
Resilient Clouds, like CRASH, is

- Willing to redesign hosts, networks and distributed systems architectures to gain security and robustness
- Concerned with Immunity, Adaptation & Diversity
- Intended to respond to threats at internet speed

But Resilient Clouds is uniquely concerned with Clouds and other ensembles of hosts, connected by networks, acting in concert to achieve enterprise goals.

Resilient Clouds will augment network *protocols* to provide visibility and adaptability.
The objective is to sustain mission effectiveness. Different mission components have different security needs and will make different trade-offs at run-time between these, quality of service, and even correctness.

\[ \sum_i P_i \sum_j w_{ij} * v_{ij} \]

- \( P_i \) = Priority of Task \( i \)
- \( W_{ij} \) = Importance of property \( j \) to Task \( i \)
- \( V_{ij} \) = Degree to which property \( j \) is achieved for Task \( i \)

**Metric for Mission Effectiveness**

**Mission Effectiveness**

- Conventional System (Crashes)
- Initial Operational Capability
- 100% Critical Functionality

**Theoretical Optimal**

**Effectiveness**

Effectiveness vs. Time

- (attack)

Approved for Public Release, Distribution Unlimited
Network Optimization - Nash Bargaining

- Bottleneck bandwidth halved every 30 seconds due to, e.g., DoS flooding attack
- Nash Bargaining rapidly adjusts bandwidth allocations to remain consistent with priorities

Approved for Public Release, Distribution Unlimited
What if We Got A “Do-Over?”

Summary
Turn the Tables: Make The Attacker Do The Work

Innate Immunity

- Novel Hardware
- Separation OS’s
- Information Flow
- Formal Methods

Adaptive Immunity

- Policy Weaving
- Automatic Patching
- Selective Playback
- Symbiotes

Dynamic Diversity

- Compiler generated Diversity
- Algorithmic Diversity
- Instruction Set Randomization

Approved for Public Release, Distribution Unlimited
Economic Tradeoff: Workfactor & Degradation

Attacker's Work Factor
% Degradation of Mission Effectiveness

Offense Winning Zone
Defense Winning Zone

COTS
Today's Protected Systems
Goal

Approved for Public Release, Distribution Unlimited
Changing the Equation

- **Recon & penetration escalation**
  - Minimize Footprint & Cover Footsteps
  - Change Work Factor to Achieve Attack
  - Attack Succeeds
  - Improve Detection, Diagnosis & Initiation of Recovery
  - Escalate & penetrate new area
  - Increase Recovery Speed
  - Destroy resources necessary for regeneration
  - Final Recovery
  - Increase Level of Regeneration

- **Degree of Maximum Degradation**
  - Change Severity of Attack

Approved for Public Release, Distribution Unlimited
What if We Got A “Do-Over?”

Will we do it better?