**Discussion**

- We will now look at a low level approach to survivability

- There are some definite potential problems
  - During the presentation, think maliciously and identify the weaknesses.

---

**Redundancy: A Curse or Blessing?**

- Recall what we said about Redundancy:

- Recovery requirements imply Redundancy

- Three Types of Redundancy
  - Information Redundancy
    » add information
    ■ e.g. error correction, authentication, codes
  - Time Redundancy
    » repeat event in time
    ■ e.g. multiple sensor readings (of same sensor)
  - Spatial Redundancy
    » physical redundancy, local or distributed
    ■ e.g. NMR, k-of-N
Putting it back together...

- How does one combine results from redundant operations?
- Fault-Tolerant Agreement
  - From Majority Voting to Byzantine Agreement
    (started with Lamport paper)
  - Many flavors
    » Network Topology
      ■ bus, ring
    » Network Protocols
      ■ ATM, TCP/IP, multicast
    » Communication Type
      ■ symmetric, asymmetric

The BRANS Approach

- BAM = Byzantine Agreement Module
  - Survivability Cluster
An Example: DNS

- **DNS (Domain Name Service)**
  - Resolves addresses
    - snake.cs.uidaho.edu = 129.101.55.119
    - DNS server maintains database of mappings

Intruder changed DNS entry

Voting → BAM → Resolve: snake

Snake = 129.101.55.22
129.101.55.119
Agreement Requirements

- Solutions with lowest overhead are applied, e.g.
  » simple majority voting,
  » Byzantine agreement with early stopping
  » full Byzantine agreement.
- Individual critical functionalities use those solutions that minimally satisfy their agreement requirements.

Note:
in the previous example a simple majority suffices, however, if the DNS table needs to be updated, stronger agreement solutions are needed that require the 4 computers shown.

Discussion

- Lets play “Devil’s Advocate”
Systems under Attack

- How does one tell if a system is under attack?
  - IDSs?
  - How “real-time” should Real-Time be?
  - Decide on a “Level of Abstraction” to be considered.

Systems under Attack

- How can the Whittaker approach be modified to help attack recognition?
  - observing
    » dependencies
    » profiles
    » timing behavior
    » …
Systems under Attack

- We will look at two examples, one is bottom-up and the other top-down.
  - The next discussion is based on the paper

- We will compare the basic approach with the concepts of the Whittaker paper.

Objective

- Achieve Survivability of Critical Functionalities
  - ultimate goal, holy grail (very general, very difficult)
  
- “Some Attacks can be dealt with at Lowest Level”

- Standard User Environment

- Implementing Survivability Mechanism
  - at the lowest level of abstraction
  - suitable for class of attacks with distinct signatures
  - survivability handlers & response agents
**Assumptions**

- Anything is possible!  
  » and it will happen!

- Intrusions will occur sooner or later

- Mechanisms that empower can be used against you

**Standard User Environment**

- **Target System**
  - Typical desktop computer
  - Mostly operated by single individual
  - Standard applications
    » browser, email, sftp, ssh, multi-media, text processor, etc.

- **System Characteristics**
  - Low utilization!
    » linux `top` command
  - “Idle Profile” of system is surprisingly clean
Off-line and On-line Survivability

- Off-line Design Process
  - clean system environment (off-line, no applications)
  - creation of attack signature database
  - attack signatures aid in identification of critical functions
  - implementation of reactionary mechanisms
    » low level (kernel handlers)
    » high level (migratory agents)
    » a priori matching of critical functionalities with critical functions

- On-line (real-time) Protective Capabilities
  - real-time attack recognition
  - at high level
    » recognition triggers response agents
  - at kernel level
    » survivability handlers get invoked (independent of attack recognition)
System Architecture

- LAN

Currently *profiled* in kernel

Critical Functionalities in Protocol Stack

Target of Survivability Features

PIII-PC - RedHat 6.2 /2.2.16

Levels of Abstraction

- Real-time Potential

High

Log File Analysis

System Call Analysis

Kernel Signature Analysis

Hardware Assisted Analysis

Low

Level of Abstraction

Real-Time Potential
Two Layers of the Architecture

Real-time

- Low-level Event Handlers
  - Survivability handlers
  - Currently used for kernel instrumentation
  - Case study: Early Stopping Agreement

- High-level Reactionary Control
  - Implements high-level survivability features
    - e.g. filtering, patching, early warning
  - Migratory Autonomous Agent System
    - Small specialized program to perform specific task
    - Off the shelf technology, (Aglets)

Survivability Architecture Overview

- System Components

  Agent Interface
  Reactionary Control
  Signature Analysis
  Attack Signature Library
  Event Handlers

Agent Agent ... Agent
Profiles

- We view a system as a collection of profiles of its functionalities $P_i$

$$P_{sys} (\Delta t) = \sum_{i=1}^{k} P_i (\Delta t)$$

$k$ is the number of functionalities active during $\Delta t$

- Functionality Profile

$$P_i (\Delta t) = (f_1 (\Delta t), f_2 (\Delta t), \ldots, f_n (\Delta t))$$

$f_j(\Delta t)$ is the number of times identity $F_j$ has been invoked during $\Delta t$

Attack Signatures

- Atomic Attacks $A_i$
  - the smallest attack technology unit
  - e.g. a port sweep, sequence of unsuccessful login attempts

- Attack Signature $S_i$
  - the portion of a profile that is attributable to $A_i$

$$S_i (\Delta t) = (f_{\alpha(1)} (\Delta t), f_{\alpha(2)} (\Delta t), \ldots, f_{\alpha(s_i)} (\Delta t))$$

$\alpha$ is a one-to-one mapping from indices of $S_i$ to indices of the identities $F_j$ profiled

$f_j(\Delta t)$ is the number of times identity $F_j$ has been called during $\Delta t$
**Attack Signature**

- **Attack Signature over Time**
  - Example: “teardrop”
    (overlapping IP(TCP) fragments are formatted to cause reassembly crashes)

**Example “teardrop”**

![Graph showing frequency over time for a teardrop attack]

**Functions**

- $f_i(\Delta t)$
- Frequency
- Time [s]
- Functions $i$
- $t$
Real-Time Attack Recognition

- Vector Analysis
  - Profile $P_i(\Delta t)$, Idle Signature $S_0(\Delta t)$, and Attack Signature $S_1(\Delta t)$ are vectors

- “Strictly Speaking”
  - there are three possible scenarios

\[
P_{sys}(\Delta t) \supseteq S_i(\Delta t) \quad \text{possible attack}
\]
\[
P_{sys}(\Delta t) \not\subseteq S_i(\Delta t) \quad \text{attack not possible}
\]
\[
P_{sys}(\Delta t) < S_i(\Delta t) \quad \text{attack not possible}
\]

Signature Analysis

- Relationship between Signatures
  - $S_i \subseteq S_j$

- Common functions
  - $S_i \cap S_j$

- Signature Correlation

\[
C(i, j) = \frac{|S_i \cap S_j|}{\min(|S_i|, |S_j|)}
\]
**Attack Signature**

- Example “teardrop” vs. “bonk”
  - bonk: malformed IP header causes packet size violation upon reassembly
  - Note: scales differ
  - Correlation is 1.0

![Graph showing teardrop attack vs. bonk attack.](image1)

**Attack Signature**

- Example “teardrop” vs. “gewse”
  - Gewse: (DoS - attack) floods identd on port 139
  - Note: scales differ
  - Correlation is 0.54

![Graph showing teardrop attack vs. gewse attack.](image2)
Correlation

“Some things seem too good to be true”

Network Survivability Architecture

Migratory Agent Framework
Case Study “Smurf”

- “Smurf” Attack
  - DDoS (limited protection against such attack)
  - attacker:
    » sends ICMP echo packets to generate multiple replies
    » attacker claims to be victim
      ■ forges source address
    » target of echo request is
      ■ all machines in broadcast subnet
      ■ “Amplifier network”
  - victim:
    » all systems in amplifier network respond
    » victim gets flooded with unwanted ICMP echo replies

- Response Agent
  - turns on filter in router

Conclusions

- Tow-layer approach to survivability
  - off-line and on-line component

- Low layer
  - Attack signatures aid in identification of critical functionalities
  - Survivability handlers applied at kernel level
  - Signature analysis triggers response mechanism at high level
    » attack recognition does not facilitate a general IDS!

- High layer
  - Migratory Agent system
  - Response agents act as reactionary mechanisms