Outline of Talk

• A sense of self for Unix processes (Review)
  • Emphasize method rather than results
• Evolutionary innovations
• General principles and lessons learned
Background

The immunological perspective

- The problem the immune system solves for the body is (almost) the same as the problem we want computer security to solve for our computers:
  - **Detecting** unauthorized use of computers, computer viruses, etc.
  - Choosing and mounting an effective response.
- Sophisticated IDS and response
  - Detect and stop attacks automatically in real time
  - Focus on system call monitoring
The biological perspective led to a set of general design principles

- Autonomy
  - On-line, real-time automated response
- Simple and generic
  - Anomaly detection, focus on executing code
- Adaptable to changing programs and environments
- Diversity
  - Of the defense mechanism and the host itself

- Collect system-call data for normally operating programs (time series)
- Build a profile of normal behavior based on these data
- Observe more (possibly anomalous) behavior
- Treat discrepancies as anomalies
- Sana Security *Primary Response*
Building the profile

- n-gram representation
- One profile per executable
- Store in fixed size array

Profiles
- 1 training array
- 1 testing array

Heuristics

<table>
<thead>
<tr>
<th>Call</th>
<th>Position 1</th>
<th>Position 2</th>
<th>Position 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td>read, getrlimit</td>
<td>mmap</td>
<td>mmap, close</td>
</tr>
<tr>
<td>read</td>
<td>mmap</td>
<td>mmap</td>
<td>open</td>
</tr>
<tr>
<td>mmap</td>
<td>mmap, open, close</td>
<td>open, getrlimit</td>
<td>getrlimit,mmap</td>
</tr>
<tr>
<td>getrlimit</td>
<td>mmap</td>
<td>close</td>
<td></td>
</tr>
<tr>
<td>close</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Anomalies:
- open, open
- open, *, getrlimit
Measuring Anomalies

Number of Misses in Locality Frame

Position in Trace

locality frame

00110011000001111100000000001
Example: syslogd intrusion
Automated Response

• Intrusion detection incurs a cost of persistent false positives
  • Perpetual novelty
  • Legitimate normal behavior evolves over time
  • Inherent ambiguity between normal and intrusive
• Automated response often ignored because false-positives are expensive
  • Must reduce systems administration burden (rather than increasing it)
  • Must be tolerant of some false-positives
Graduated response

• Process Homeostasis (pH):
  • Computer autonomously monitors its own activities
  • Continually makes small corrections to maintain itself in a “normal” state

• Anomalous sequences trigger system-call delays
  • Exponentially increasing delay
  • Small delays imperceptible to users
  • Long delays trigger timeout mechanisms at network and application level

• HP’s ProCurve network Immunity Manager
process Homeostasis (pH)

Somayaji and Forrest Usenix, 2000
Stopping attacks in real-time

ssh Trojan program (buffer overflow)

Linux capabilities bug (via sendmail)

Note: Other ssh and sendmail processes unaffected
Mimicry Attacks

- Sequences of system calls that exploit a vulnerability but appear normal
- Relies on successful code injection
- Code bloat from nullified calls
- Mimicry has to persist as long as the attacker exploits the process
- Diversity of normal profiles is a potential barrier
- Also, non control flow attacks

Wagner and Dean CCS 2002
Evolutionary Innovations
*Many authors (see paper)*

- Data modeling methods
- Extensions
  - Data flow (sys call arguments)
  - Execution context (PC)
  - Static analysis
- Other observables
  - Library calls, JVM, HTTP requests, ...

**Capture system call trace:**

..., open, read, `mmap`, `mmap`, `open`, `getrlimit`, close, ...

**Extract sequences:**

- n-grams
  - `mmap`, `mmap`, `open`, `getrlimit`  
  - `mmap`, `open`, `getrlimit`, close

**Data Modelling**

- `open`, `getrlimit`  
  - `mmap`, `*`, `getrlimit`  
  - `mmap`, `*`, `*`, `getrlimit`  
  - `getrlimit`, `close`  
  - `open`, `*`, `close`  
  - `mmap`, `*`, `*`, `close`

**lookahead pairs**

**DFA, HMMs**
The biological analogy led to a set of general principles

- Generic
  - Universal weak methods are applicable to many problems
  - Do not require specialized domain knowledge
  - Coverage of a broad range of attacks, but not 100% provably secure
The biological analogy led to a set of General principles

- Generic
- Adaptable
  - To changes in the environment and self
  - Simple learning to construct models and update over time
The biological analogy led to a set of General principles

- Generic
- Adaptable
- Autonomy
  - Graduated response
  - Need for speed dictated simplicity
The biological analogy led to a set of General principles

- Generic
- Adaptable
- Autonomy
- Diversity

- Each profile is unique, making it difficult for the attacker to predict the profile
- Led to automated diversity project
Lessons Learned

- Designed repeatable experiments
  - Open source code and data
  - Comprehensible system design that focused on one hypothesis
- Careful comparison between methods is difficult
  - Environments are complex and systems difficult to replicate
  - Metrics emphasize breadth of coverage and corner cases
  - Results depend heavily on data set choice; methods might not matter
Conclusion
Engineering practices based on biology

• Why do we need them?
  • Evolution of the software ecosystem (software rot, malware)
  • Dynamic, mobile, complex, and hostile environments
  • Moore’s Law won’t rescue us

• Hallmarks
  • Simple and generic
  • Computationally and memory efficient
  • Automatically self-tuning, distributable, diverse, and autonomous
What I’m doing now

• Autonomous security for autonomous systems (BGP), privacy enhancing data representations (Negative Databases)

• A scaling theory for the rest of computer science

• Using GP to fix bugs in software automatically
Biological defense mechanisms
Applied to computation

• **Immunology:**
  - Protect an individual (single host or a network) against network epidemics and other forms of attack.
  - Antivirus programs, intrusion-detection systems
  - Sana Security *Primary Response*

• **Autonomic responses, e.g., homeostasis:**
  - Tightly coupled low-level detection/response phases.
  - pH and network (virus) throttling.
  - *HP’s Virus Throttle*
Biological defense mechanisms
Applied to computation cont.

- **Diversity:**
  - Genetic diversity leads to population-level robustness.
  - Disrupt software monoculture using randomization and/or evolution.
    - *Microsoft Vista Address Space Randomization*

- **Epidemiology:**
  - Network-based control of viruses/worms.
  - Focus on network topology (the epidemic threshold).
  - Survivability and attack resistance (*PGBGP---work in progress*)
Other biological defense mechanisms
Still to be tapped

- The innate immune system
- Ecological interactions and evolutionary biology
  - Malware ecology: Malware interactions, indicator species, etc.
  - Automated bug repair using evolutionary methods
  - Optimal levels of defense in depth
- Intracellular defenses and repair mechanisms
  - RNAi
  - Restriction enzymes
Significance

• Early successful example of anomaly intrusion detection

• On-line, real-time, adaptive, automated response
  • Stops attacks in real-time

• Diversity of protection

• Sana Security started by former UNM student, Steven Hofmeyr


Mantra

• The only code that can hurt you is code that actually runs

• Keep it simple stupid (KISS)

• Never let the geeks forget there is a bigger picture

• Nothing says it won’t work