Bio-Inspired Security

CSC 790

Wake Forest University
Department of Computer Science

Spring 2014

What makes cyber security a difficult problem?

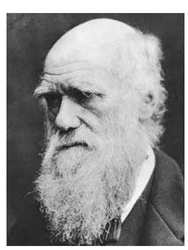
- Mathematical approaches often seek optimal
- Traditional methods need well defined problems
- Security problems are often ill-conditioned
  - Many steps and inputs may be unknown
- Security problems are adversarial, more difficult

Natural Systems and Ill-Conditioned Problems

- Do not strive for optimal, just try to be good enough
- If a situation changes... no problem we can adapt
- Multi-stability allows coexistence of many stable states
- Robust, tolerant of mistakes, perhaps *learning*
- Scaleable

Nature-Inspired Security Research @ Wake

swarm defense



evolving computers

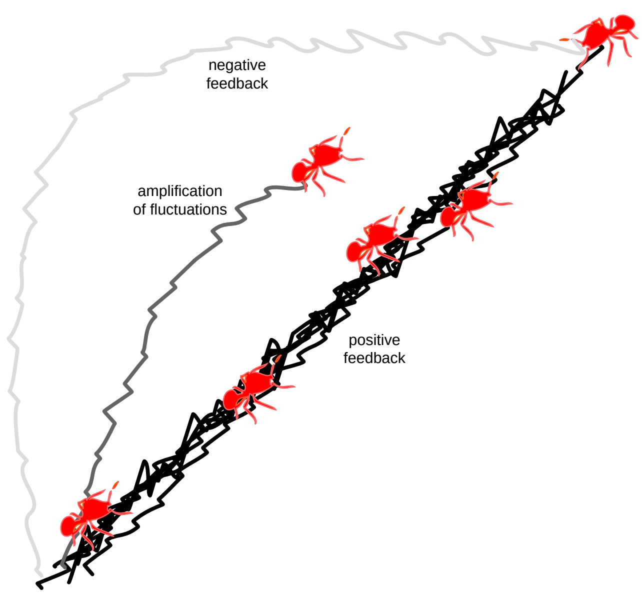
watching for patterns

Swarm Intelligence

One interesting natural system behavior is swarm intelligence
"Emergent collective intelligence of groups of simple agents."

- Multiple agents seek local goals, collectively benefit group
  - Complex, global solutions emerge from local solutions
  - Ant colony, bee hive, water drops
- Principles applied to other difficult problems
  - Dynamic routing, traveling salesman, airplanes, ...
- There are a few swarm attributes that make this work

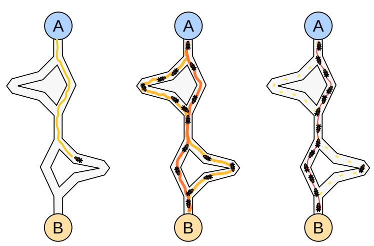
Swarm Attributes



Simple Ant Colony

- Assume only one type of ant agent
- Ants wander seeking food (amplification of fluctuation)
  - If successful, return leaving pheromone trail
- Pheromone is a chemical that dissipates over time (negative feedback)
  - Pheromone strength influences wandering of other ants
  - Stronger the pheromone, more likely ant follows (positive feedback)
- Can be used to solve *difficult problems*

Finding the Best Path



- Find the shortest path between two points, A (home) and B (food)?
- Ants will randomly wander until food is found
  - Multiple paths may exist
  - As ants return, add more pheromone, reenforcing path strength
  - Shortest path should have the strongest pheromone
- Can be used to solve other similar, difficult problems
  - Internet routing, traveling salesman, security, global warming, ...

CID/Digital-Ants

- Digital ants uses ant colony properties to discover an intrusion
  - Positive feedback - pheromone attract others (help solve problem)
  - Negative feedback - pheromone evaporates
  - Amplification of fluctuation - ants always, more or less, wander
- As a results, the approach is scalable and robust
  - Perhaps not as quick and standard/traditional approaches...

Biological Immune Systems

- Antigens (foreign proteins) are recognized by antibodies (detectors)
  - Antibodies are highly specific, only bind to certain antigens
  - If they do bind, events cause the antigen to be destroyed
- Antibody are covered with antigen detectors
  - What they detect, they destroy, hopefully not good cells...
  - Therefore must be able to discriminate between *self* and *non-self*
- Alternative view of detection (or how it could work)
  - Antibody responds to damaged cells, since antigens cause damage

Self and Non-Self

- Two basic approaches for detection
  - Know self, anything different is non-self
  - Know non-self, anything different is self
- Antibodies know non-self (similar to signature-based IDS)
  - Kind of counter-intuitive given there are $\approx 10^{15}$ antigens and $\approx 10^{6}$ human proteins
  - There are $\approx 10^{10}$ antibodies, of which there are $\approx 10^{7}$ types
  - Therefore, there are more antibodies than proteins, but fewer than antigens

  Can an immune approach be applied to computer systems?

Computer Immune System

- Possible to make the analogy of self/non-self for computers
  - Self is a set of strings and detectors are a set of strings
- String binding (detection) is done using matching function
  - Simple approach is r-contiguous bits, which returns true when two strings match in more than r contiguous positions
- Immune inspired approach creates detectors for abnormal (non-self)
  - Therefore, generate strings that do not match self (normal)
  - If something matches a string, then it's considered non-self

  Would this method result in higher or lower false positives? How many non-self detectors (strings) are necessary?

Immunization Performance

- Probability of false negatives (p_f) will go down as the number of detectors increases (n_r)
  - In addition the number of detectors required will depend on how many (malware) strings each can detect
- Let p_m be the probability that two random strings will match
  - False negative error rate is approximately

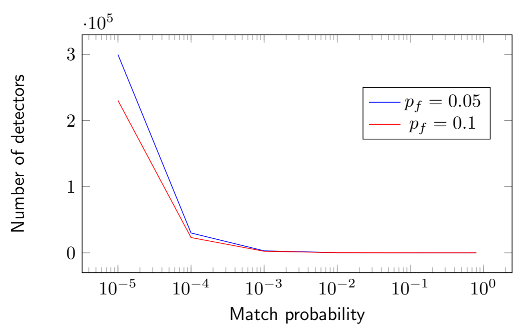
$$p_f = (1 - p_m)^{n_r} \approx e^{-p_m n_r}$$

which can be rearranged as

$$n_r \approx \frac{-\ln p_f}{p_m}$$

  - Therefore the if p_m is small, then n_r will be large

m	r	l	p_m
2	8	64	0.108
2	8	128	0.215
2	16	32	0.00013
2	16	64	0.0008



- Seems as if a larger p_m is desirable, however p_m depends on
  - Number of alphabet symbols (m), number of symbols in the string (l), and number of contiguous matches (r)

$$p_m \approx m^{-r} \frac{(l - r)(m - 1)}{m + 1}$$

  - These variables must be set as to sufficiently define self/non-self, for realistic applications p_m will be small
- See `http://www.cs.unm.edu/~immsec/publications/virus.pdf`

Creating Detectors

- Originally, detectors were created at random
  - String compared to self
  - If no match then the string was kept
- Number of detectors that need to be generated, n_{ro}

$$n_{ro} \approx n_r^{p_m n_s}$$

  where n_s is the self set size
  - As a result, the detector discovery process may take awhile

Some Immunization Alternatives

- Number of detectors can be reduced, if signatures are more complex
  - Signature based scanning, where signature also detects action
  - Developing signatures is more difficult (required an expert)
- MBI (*yes, that's sdrawkcab*) developed a system that had *decoy* programs to help automate the signature process
  - Trap viruses and keep information to create signatures
  - Honeypot-like, but at the process level

Diversity

- Biological systems often use diversity as a defense
  - Making individuals slightly different (genetically and through experience) overall species *should* be more robust
  - This idea is counter to the monoculture is computer systems
- Instead of removing vulnerabilities, make each computer vulnerable in a different way
  - However, there is a limit to how diverse a program can be
- Stack randomization (discussed in our MT section), is an example
  - Vulnerability remains, but the exploit is unique to each execution
- Another diversity technique attempts to obfuscate code
  - Have multiple definitions for components that are randomly selected at compile time