Evolutionary Based Moving Target Cyber Defense

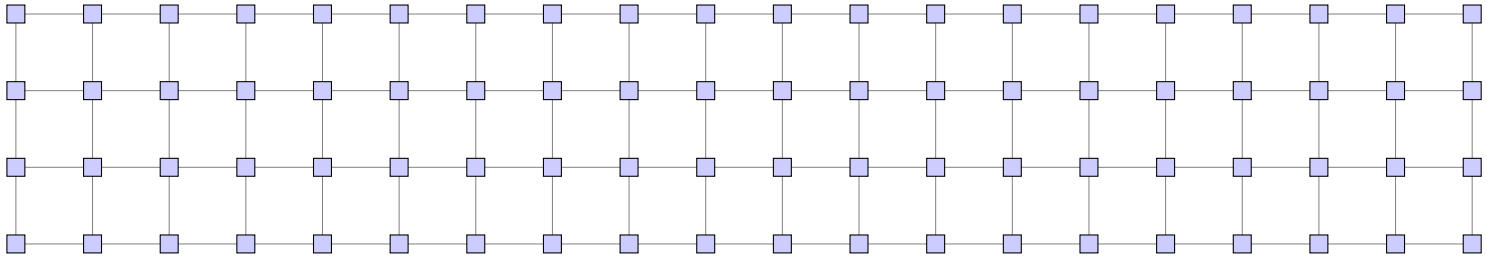
David J. John, Robert W. Smith, William H. Turkett
Daniel Cañas *and* **Errin W. Fulp**

WAKE FOREST
UNIVERSITY
Department of Computer Science

National Science Foundation
Secure and Trustworthy Cyberspace
Award 1252551

July 13, 2014
Bastille Day Eve

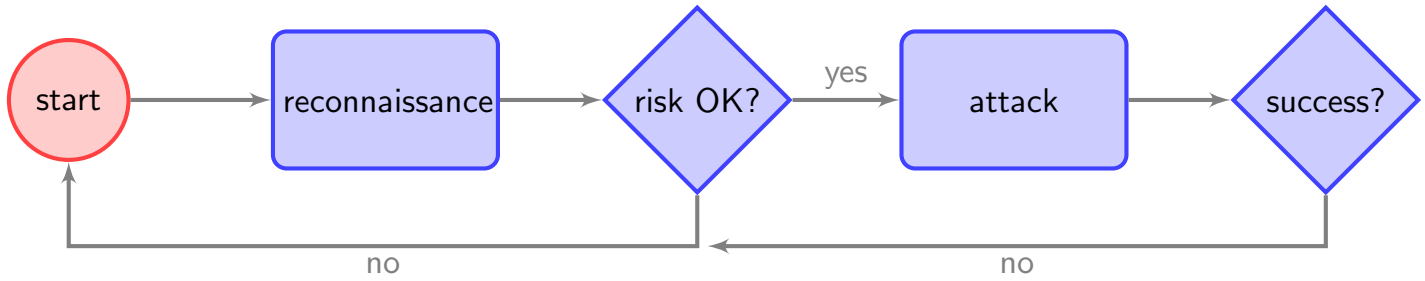
Computer Infrastructure Defenses



- Computers are often deployed in a deterministic fashion
  - Cluster computing, office computing, VMs, etc...
  - Often done to simplify life-cycle management
- Unfortunately a predictable environment is great for attackers
  - Exploit for one computer is applicable to all
- One type of defense is to create a **moving target** environment

Moving Target Environment

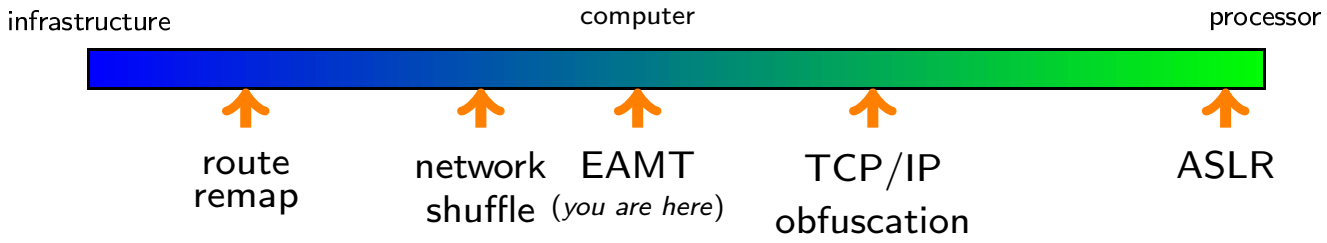
Abbreviated attack process



- Most cyber attacks start with a reconnaissance phase
  - Gather intelligence about potential targets
  - Can require substantial time and effort
- Moving target environments make reconnaissance ineffective
  - Defended system changes over time
  - Provide security through diversity

Moving Target Examples

Categories of MT environments



- Several successful examples of moving target environments
  - Memory address randomization (ASLR)
  - Network address shuffling
  - Routing remapping
- Want to create **a moving target defense for computers**
  - *Change properties of the system*, do not hide or cloak
  - *Hopefully* evolve to more secure systems

Configurations and Moving Targets

```
⋮
# Create "/keygen" if it doesn't exist.
class gen_class
  {
    file { "/etc/cipher/keygen.lst"}:
       ensure => present,
       mode => 644,
       owner => root,
       group => root

  }
}
⋮
```

```
⋮
NameVirtualHost 172.20.30.40
<VirtualHost 172.20.30.40>
# primary vhost
DocumentRoot /www/subdomain
RewriteEngine On
RewriteRule /.* /www/subdomain/index.html
ServerSignature Off
ServerTokens Prod
# end definition
</VirtualHost>

⋮
```

```
⋮
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
# increase Linux autotuning TCP buffer limits
# min, default, and max number of bytes to use
# (only change 3rd value, make it 16 MB or more)
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
# recommended to increase this for 10G NICS
net.core.netdev_max_backlog = 30000
net.ipv4.tcp_congestion_control=cubic
# are you really reading this?
⋮
```

- Attacker observes *computer properties* during reconnaissance phase
  - Configurations define properties of the OS and applications
  - Changing the configuration could create a moving target (*which is different than masking*)
- Defining one good configuration is difficult
  - Windows system typically has over 200,000 registry entries
  - *We want to find multiple good configurations...*

Difficulty of the Problem

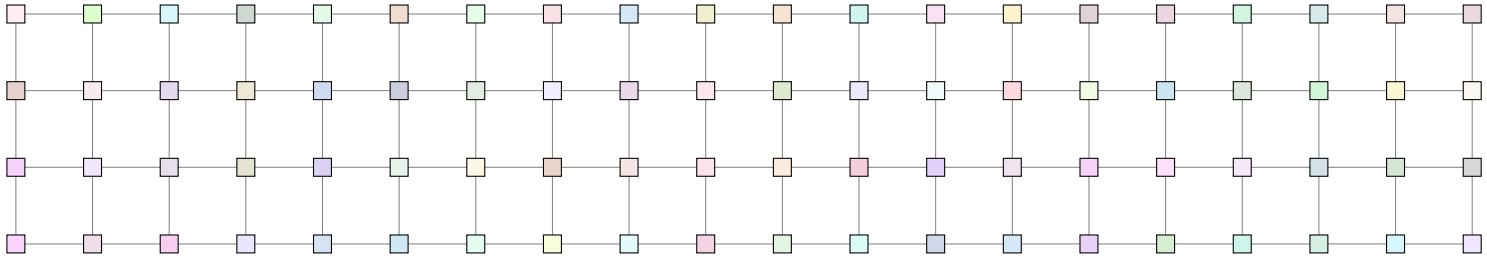
- Let C be a configuration of n individual settings, $C = \{p_1, p_2, ..., p_n\}$

Description	Possible Values
`KeepAlive`, allow requests over the same connection	0, 1
`KeepAliveTimeout`, wait time to wait for requests	0 - 1800
`Indexes`, automatic directory indexing	0, 1
`LimitRequestBody`, limit the message size	0 - 65535
`LimitRequestFields`, limit number of HTTP requests	1, 0
`LimitRequestFieldSize`, limit HTTP header field size	0 - 32767

$$C = \{1, 800, 1, 32767, 0, 1\}$$

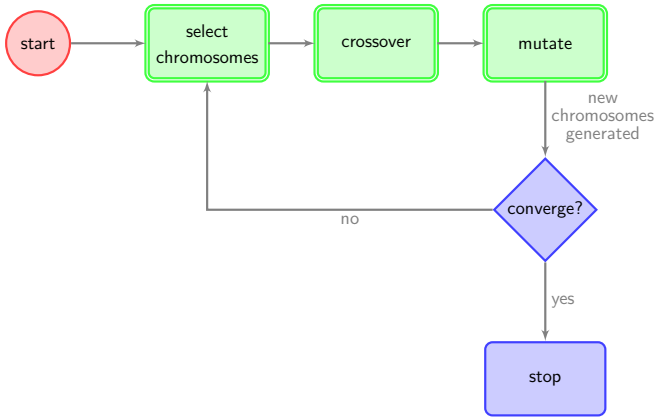
- Parameters may be interdependent, forming *parameter chains*
  - Certain subset of settings may be required for feasibility or security
  - Setting Apache `KeepAliveTimeout` and `StartServers` too high can enable a DoS attack
- Can model the configuration as a Boolean expression
  - Dependencies can be expressed using AND, OR, and NOT
  - Finding a secure configuration is similar to the satisfiability problem

Moving Target Objectives



- Want configurations to be diverse, temporally and spatially
  - **Temporally** - difference in a single computer over time
  - **Spatially** - difference between computers at any point in time
- *Difficult, but is it possible to search for multiple secure configurations?*
  - Perhaps allowing the system to adapt to changing conditions

Evolutionary Algorithms



- Evolutionary Algorithms (EAs) mimic evolution to find solutions
  - Better solutions are created from good solutions
  - Randomness of the process can be used to create diversity
- Applying an EA for configuration discovery
  - Model configuration as a chromosome
  - Define chromosome fitness (ranking)
  - Develop appropriate selection, crossover, and mutation processes

Modeling Configurations as Chromosomes

- Configuration consists of multiple parameters
  - Individual configuration parameters are the chromosome traits
  - Therefore configuration is a chromosome, C

Name	Description	Value	Purpose
file permission	permissions for the *secret* file	222	security
login banner	message displayed upon login	2	diversity
file ownership	ownership of the *secret* file	774	security
max open files	change max number of open files	220	diversity
max file	change max file descriptor	409	diversity
⋮	⋮	⋮	⋮

$$C = \{222, 2, 774, 220, 409, \ldots\}$$

- Chromosome may have *security* and/or *diversity* parameters
  - Security parameters affect the security of the system
  - Diversity parameters affect something observable in the system
  - All chromosomes must represent **feasible** configurations
- Chromosomes will be be ranked, need a measure of *fitness*

Configuration Security

- Fitness of a chromosome based on the attacks encountered
  - Attack difficulty and impact
- Common Vulnerability Scoring System (CVSS) is a 6 part vector

	Part	Description	Possible Values
Difficulty	AV	access vector	Local, Adjacent-network, Network-external
	AC	access complexity	High, Medium, Low
	Au	authentication required	Multiple, Single, None
Damage	C	confidentiality	None, Partial, Complete
	I	integrity	None, Partial, Complete
	A	availability	None, Partial, Complete

CVE	CVSS Vector	CVSS Score
Default `ssh` password	AV:N/AC:L/Au:N/C:P/I:P/A:P	7.2 (*high*)

  - Forms the basis for EA fitness

Chromosome/Configuration Fitness

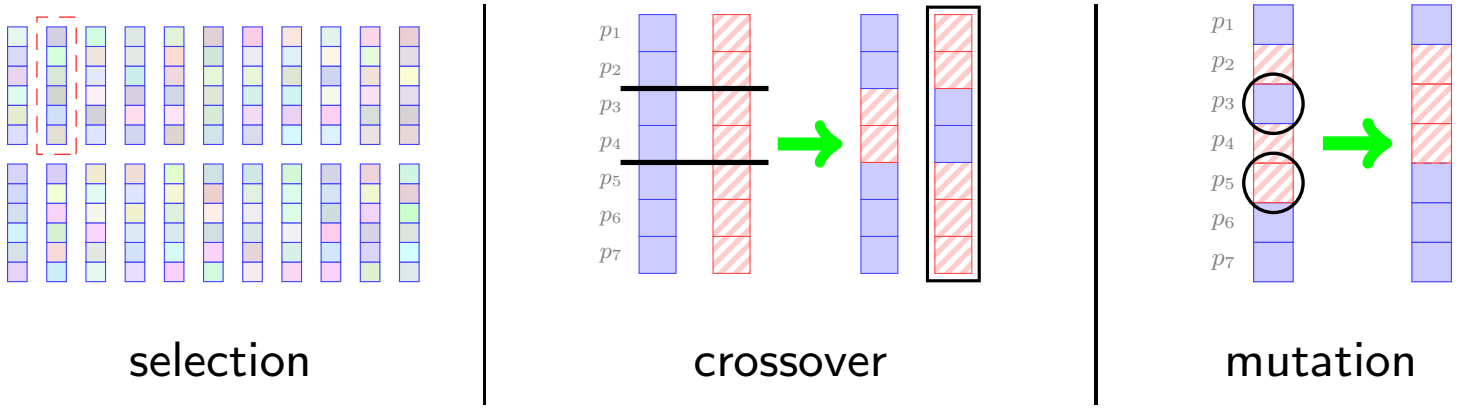
- Three levels used to score each CVSS vector part

AV/AC/Au/C/I/A

Score	Example CIA Description	Value
Good	Suffered no attack	100
Medium	Attacked with minimal impact	10
Bad	Attacked with maximum impact	1

  - Therefore best parameter score is 600, the worst is 6
  - Parameter scores added to provide a configuration score
- Chromosome fitness value will not always be correct
  - No incidences does not necessarily indicate a secure configuration
  - New vulnerabilities will be discovered

Evolutionary Algorithm Processes



- Selection identifies parents for new chromosomes from current pool
  - Roulette wheel selection, chromosomes weighted based on fitness
  - Prefers chromosomes with higher fitnesses
- Crossover combines selected chromosomes to create new chromosomes
  - Two point crossover used, creates new offspring (configuration)

Mutation

- Randomly change offspring parameters based on probability p_m
  - Provides diversity and exploration of solution space
  - Can mutate the parameter value or the domain
- Parameter Value Mutation (PVM)
  - For a parameter, randomly select among all possible values

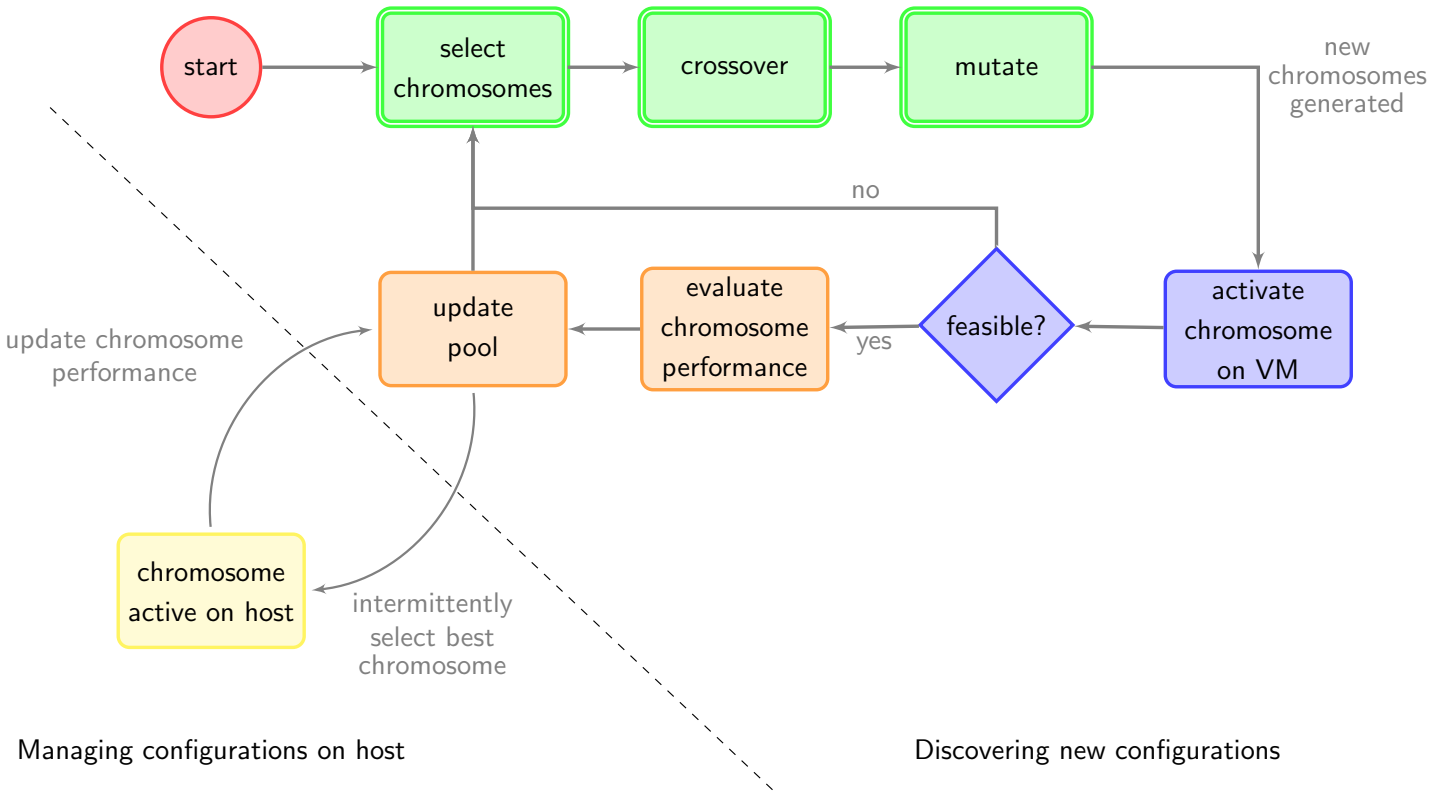
{GET, POST, PUT, DELETE, CONNECT, PATCH, OPTIONS, LOCK, UNLOCK}

- Parameter Domain Mutation (PDM)
  - Modifies the parameter domain (set of possible values)
  - If a setting is deemed "poor" it is excluded (removes bad choices)

{GET, POST, PUT, DELETE, CONNECT, PATCH, OPTIONS, LOCK, UNLOCK}

  - Randomly select from remaining values

EA as a Moving Target Defense

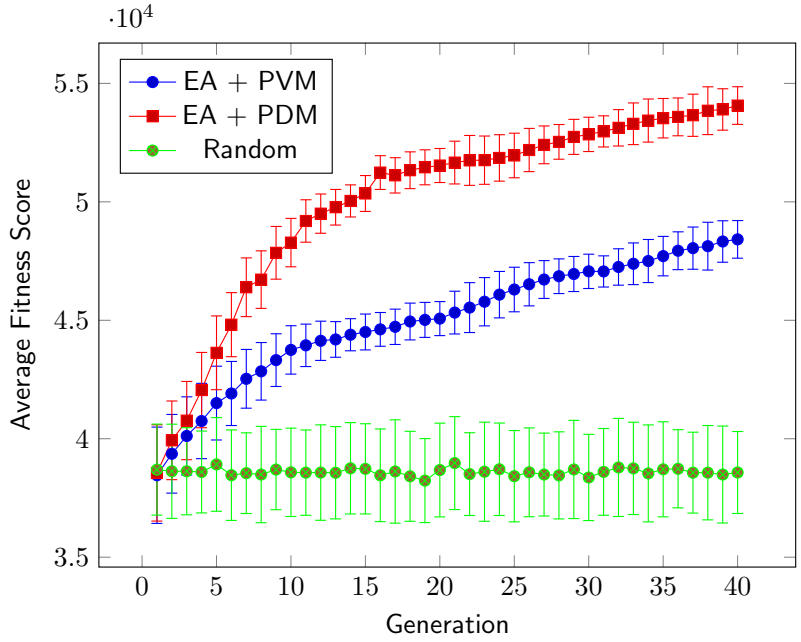


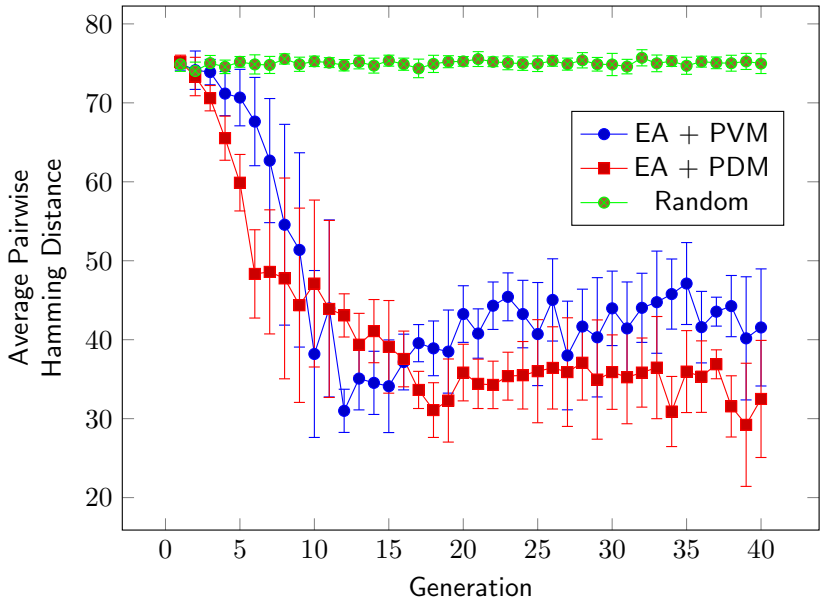
- System does not converge to a single value, it continually iterates
  - Potentially adapt to changing conditions

Experimental Results

- Experiments using 140 RHEL5 and Apache 2.2 installed servers
  - Managed configurations have 102 parameters, initially insecure
  - Parameters have different possible value ranges
- EA+PVM and EA+PDM compared with a random search algorithm
- Performance based on security and diversity (spatial) provided
  - Fitness score *based on* parameter settings (*CVSS-like*)
  - Diversity is the sampled Hamming distance between configurations
  - Seek high fitness and diversity values

EAMT Results

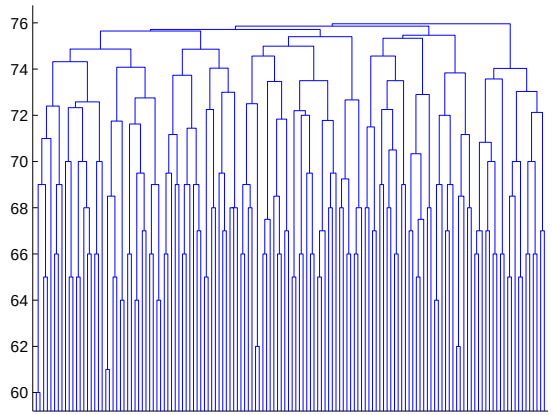




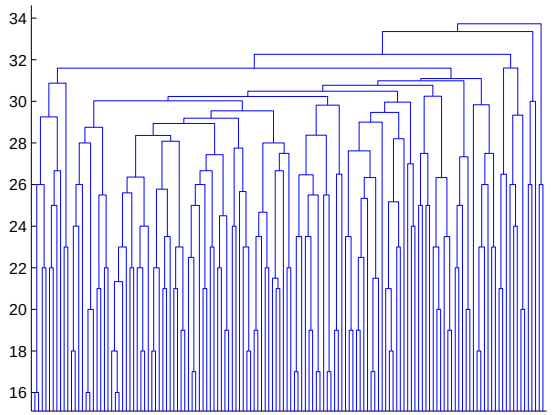
- Both EA variations were able to discover secure configurations
- GA + PDM has better fitness, but less diversity
  - PDM reduces the alternatives (bad settings) per parameter

Initially Diversity

Configurations

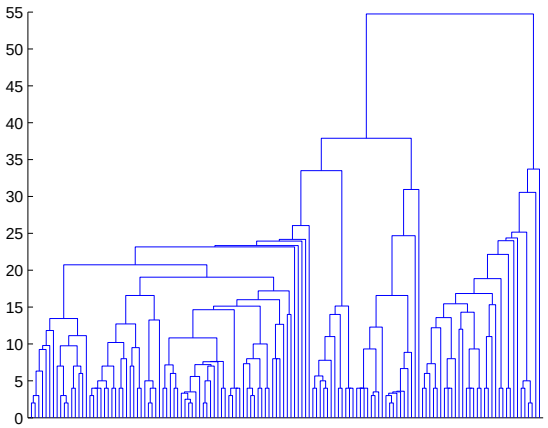
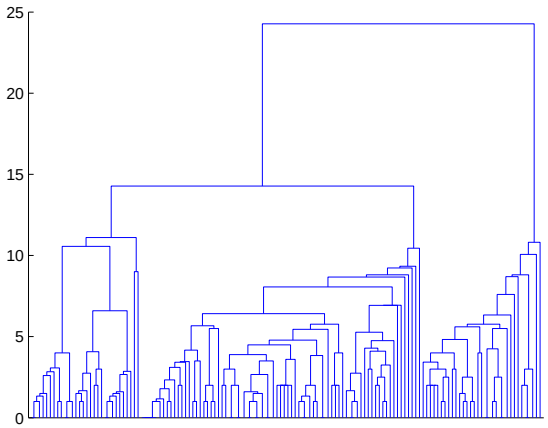
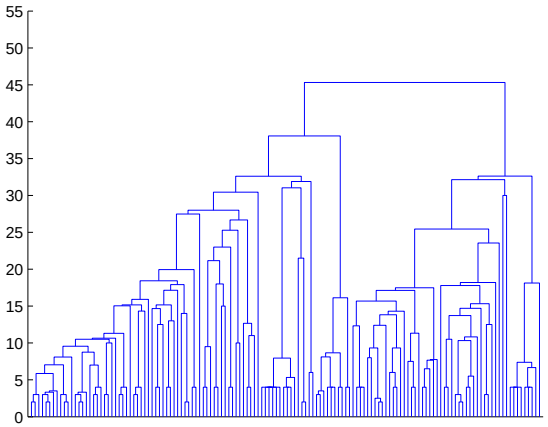
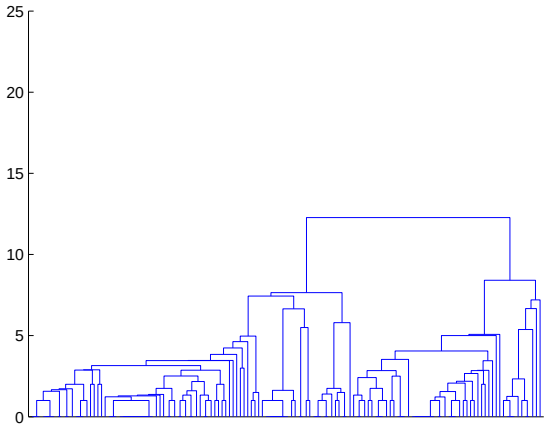


Vulnerabilities

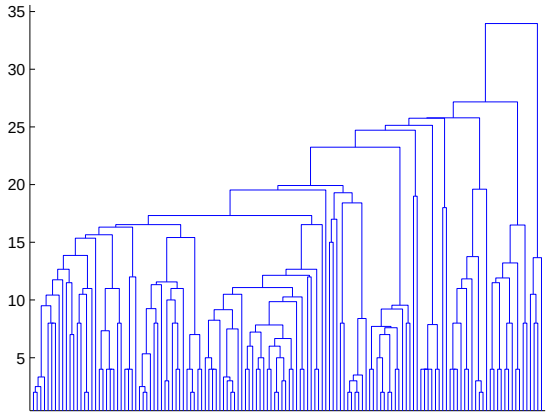
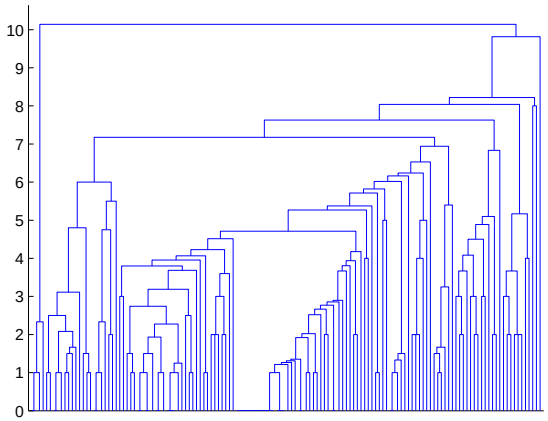
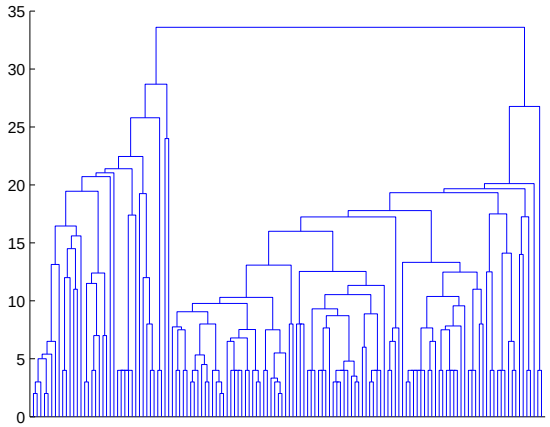
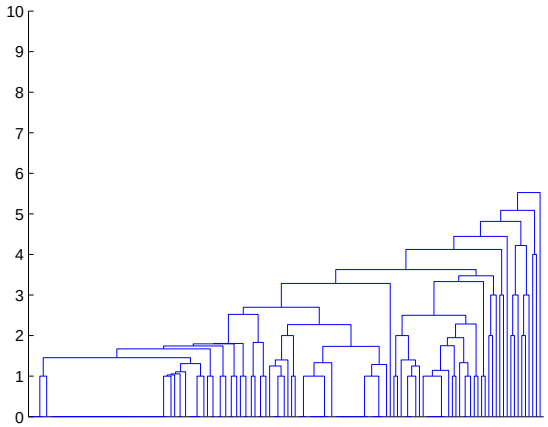


- Configuration plot indicates diversity of the configurations
- Vulnerabilities plot indicates diversity of the vulnerabilities

Diversity at Generation 21

	Configurations	Vulnerabilities
PVM		
PDM		

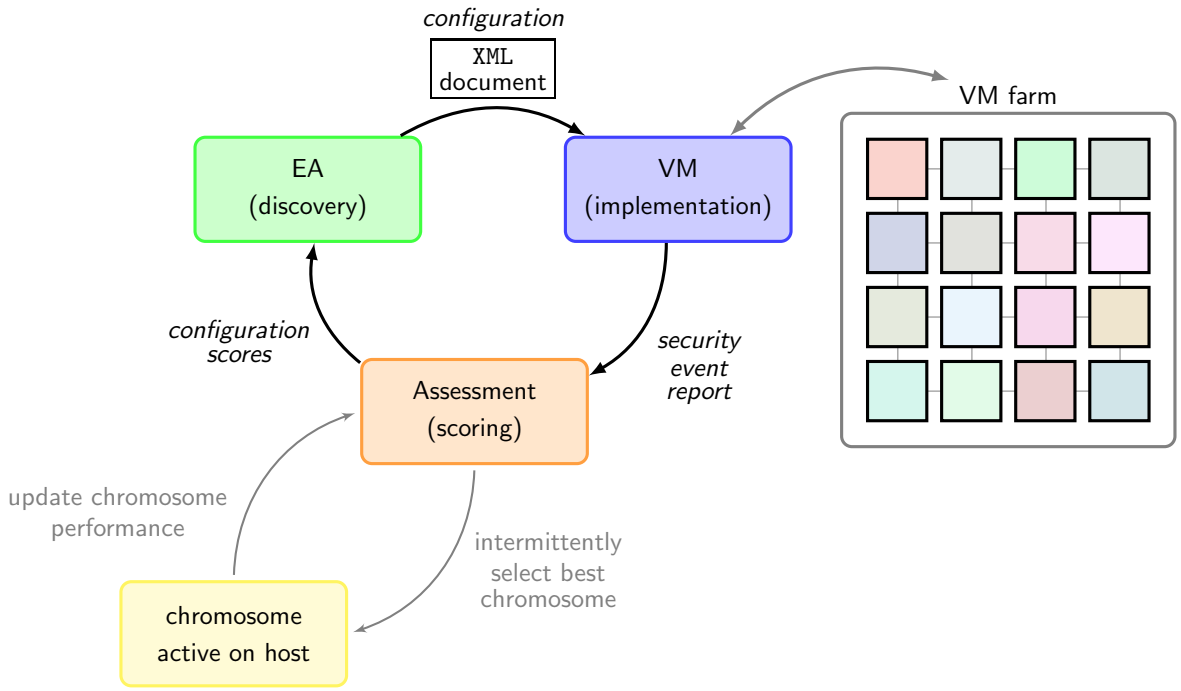
Diversity at Generation 41

	Configurations	Vulnerabilities
PVM		
PDM		

Conclusions and Future Work

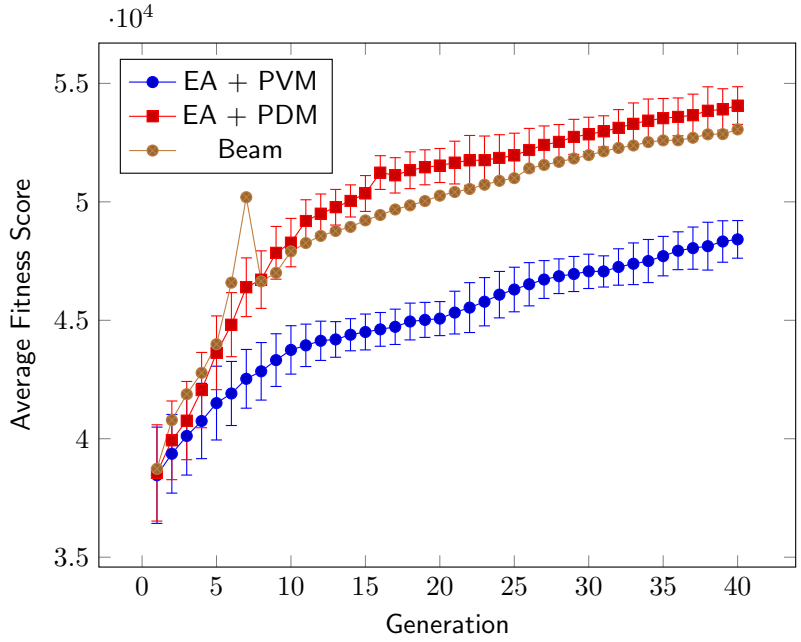
- Moving Target Defense (MTD) attempts to invalidate reconnaissance
  - Changing computer configurations is one approach
- Evolutionary-Algorithms (EA) can find computer configurations
  - Provide randomness, which provides necessary diversity
  - Initial results show the approach has promise with web servers
- Need to consider more implementation details
  - Prioritize attacks based on CIA scores
  - Detection of successful defenses
  - Improving PDM with probabilities
  - Adding functional testing

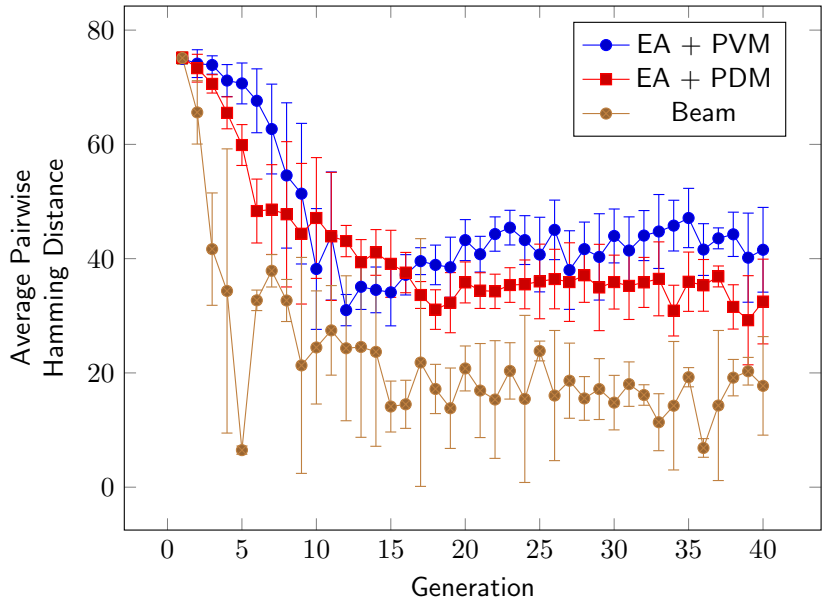
EAMT Framework Operation



- Three components continually iterate
  - Configuration database (chromosome pool) should improve
- Periodically, chromosome selected and instantiated on actual server
  - This provides a moving target defense for the actual server

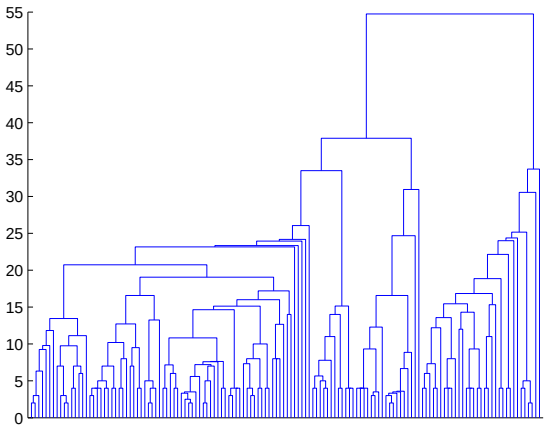
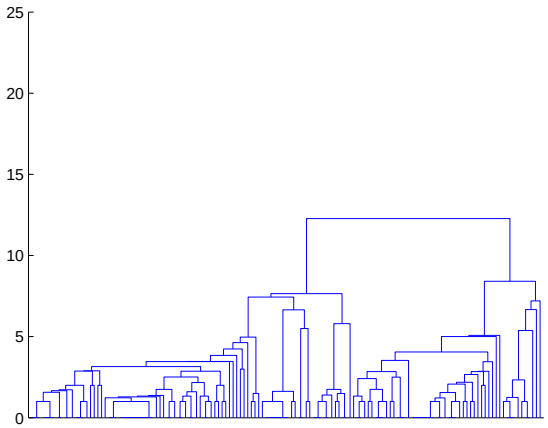
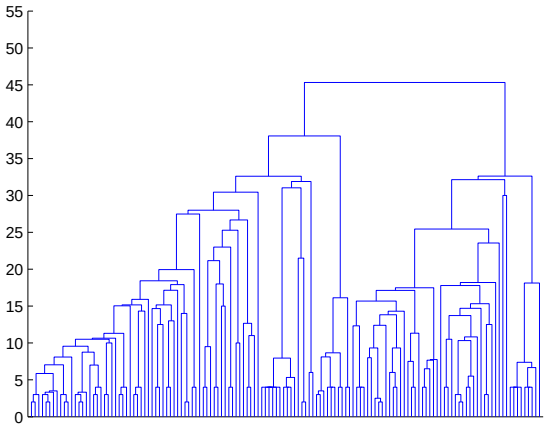
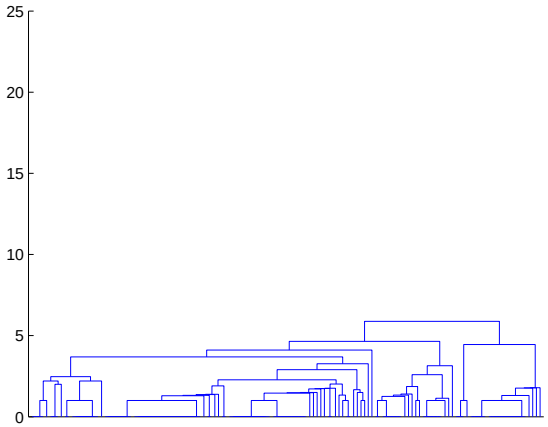
EAMT and Beam Search Results



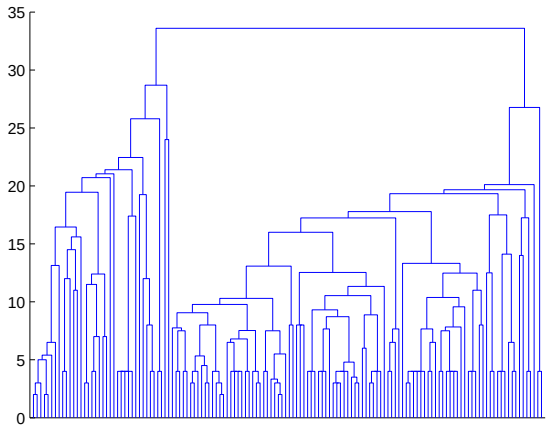
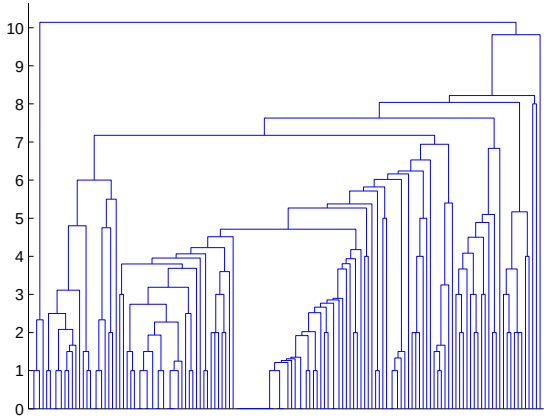
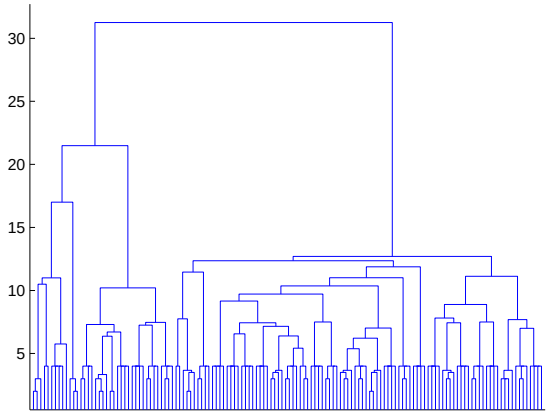
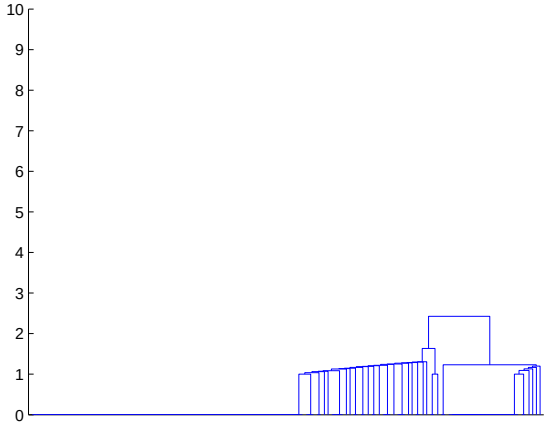


- Beam search operates by expanding the most promising solutions from the current set of alternatives
  - As a result, diversity is very low

Diversity at Generation 21

	Configurations	Vulnerabilities
PDM		
Beam		

Diversity at Generation 41

	Configurations	Vulnerabilities
PDM		
Beam		

Initial Framework Experiment Configuration

Parameter	Value Type	Description
`.htpasswd`	Binary	Allow configuration changes on a per-directory basis
`ServerTokens`	Value from a list	Controls the information about server OS and modules.
`KeepAlive`	Binary	Allow multiple requests to be sent over the same connection.
`KeepAliveTimeout`	Positive integer	Time to wait for a subsequent request before closing the connection.
`FollowSymLinks`	Binary	Allow symbolic links in a directory to be followed.
`IncludesNoExec`	Binary	Allow server side include execution.
`Indexes`	Binary	Allow automatic directory indexing.
`LimitRequestBody`	Positive integer	Limit the size of message body.
`LimitRequestFields`	Positive integer	Limit number of request HTTP header fields allowed.
`LimitRequestFieldSize`	Positive integer	Limit the size of an HTTP request header field.
`LimitRequestLine`	Positive integer	Limit the size of a client's HTTP request-line.
`autoindex_module`	Binary	Allow icons for directory listings.
`AllowOverride`	Value from a list	Controls if earlier configuration directives can be overridden.
`LimitExcept`	Value from a list	Limit request methods.