Architecture-Based Self-Protection: Composing and Reasoning about Denial-of-Service Mitigations
Context and Motivation

- Modern software systems operate in **constantly changing environments**
  - **Security**: constant appearance of new threats, vulnerabilities

- Current approaches to self-protection
  - Agnostic to system specifics
  - Threat-specific
  - Ignore **business context**
  - Application-level approaches often designed as part of the system
Architecture-Based Self-Protection

- **Architecture-based self-adaptation** has addressed these issues in the context of other properties (e.g., performance, cost)

- **Architecture-based self-protection**
  - Separates concern of protection into a control layer
  - Uses **architecture models** as a basis for **reasoning about detection and mitigation**
  - Allows reasoning about **security in the context of other business properties**
  - Promotes reuse of threat detection and self-protection strategies across systems

In this Talk

- Formal reasoning about the composition of mitigation approaches

  - *When to apply particular tactics and why*
    - Impact of security tactics on other system qualities
    - Composing security tactics into strategies

  - **Context-sensitive strategy selection**
    - Utility theory

  - **Analysis of the state space**
    - *Which* strategies get selected *when*
    - Effect of strategies on system utility
Outline

- Architecture-based self-adaptation in Rainbow
- **Example:** Denial of Service in Znn
- Architecture-based self-protection in Rainbow
- Validating the strategy space
  - Strategy selection analysis
  - Strategy impact analysis
- Conclusions and future work
Rainbow Approach

- A framework that
  - Allows one to add a **control layer** to existing systems
  - Uses **architecture models** to detect problems and reason about repair
  - Can be **tailored to specific domains**
  - Separates concerns through **multiple extension points**: probes, actuators, models, fault detection, repair

- A language (Stitch) for programming repair actions
  - **Tactic** – primitive adaptation step
  - **Strategy** – decision tree for tactic execution
Rainbow Framework Overview
Stitch: A Language for Specifying Self-Adaptation Strategies

- **Control-system model**: Selection of next action in a strategy depends on observed effects of previous action.
- **Value system**: Utility-based selection of best strategy allows context-sensitive adaptation.
- **Asynchrony**: Explicit timing delays capture “settling time”.
- **Uncertainty**: Effect of a given tactic/strategy is known only within some probability.

```plaintext
strategy Challenge [unhandledMalicious || unhandledSuspicious] {
  t0: (cNotChallenging) -> addCaptcha () @[5000] {
    t0a: (success) -> done;
    t0b: (default) -> fail;
  }
  t1: (lcNotChallenging) -> forceReauthentication () @[5000] {
    t1a: (success) -> done;
    t1b: (default) -> fail;
  }
}
```
Example: Denial of Service in Znn

- **Typical news website infrastructure**
  - Pool of replicated servers connected to load balancer
  - Size can be dynamically adjusted
  - Servers can deliver contents with different fidelity levels (text, images, videos...)
  - Content fidelity can be dynamically changed

- **Application layer DoS** (e.g., Slowloris)

- **Quality objectives**
  - Performance: request–response time for legitimate clients
  - Cost: number of active servers
  - Maliciousness: percentage of malicious clients
  - Annoyance: disruptive side effects of tactics
Tactics and Strategies

- **DoS mitigation tactics/strategies selected to provide interesting analytical situations**
  - For example, Adding capacity is much less aggressive than Blackholing, but it is more costly

<table>
<thead>
<tr>
<th>Tactic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add capacity:</td>
<td>Activate additional servers to distribute the workload</td>
</tr>
<tr>
<td>Blackhole</td>
<td>Blacklists clients, requests are dropped</td>
</tr>
<tr>
<td>Reduce service</td>
<td>Reduce content fidelity level (e.g., text vs. images)</td>
</tr>
<tr>
<td>Throttle</td>
<td>Limits the rate of requests accepted</td>
</tr>
<tr>
<td>Captcha</td>
<td>Forward requests to captcha processor to verify that the requester is human</td>
</tr>
<tr>
<td>Reauthenticate</td>
<td>Forces clients to reauthenticate</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outgun/Absorb</td>
<td>Combines Add capacity and Reduce service</td>
</tr>
<tr>
<td>Eliminate</td>
<td>Combines Blackholing and Throttling</td>
</tr>
<tr>
<td>Challenge</td>
<td>Combines Captcha and Reauthenticate</td>
</tr>
</tbody>
</table>
Tactics and Strategies

```plaintext
tactic addCaptcha () {
  condition {exists lb:D.ZNewsLB in M.components | lb.captchaEnabled;}
  action {
    let lbs = {select i : D.ZNewsLB in M.components | i.captchaEnabled;};
    for (D.ZNewsLB l : lbs) {
      M.setCaptchaEnabled (l, true);
    }
  }
  effect (forall lb:D.ZNewsLB in M.components | lb.captchaEnabled;)
}

strategy Challenge [unhandledMalicious || unhandledSuspicious] {
  t0: (cNotChallenging) -> addCaptcha () @[5000] {
    t0a: (success) -> done;
    t0b: (default) -> fail;
  };
  t1: (lCNotChallenging) -> forceReauthentication () @[5000] {
    t1a: (success) -> done;
    t1b: (default) -> fail;
  }
}
```
Strategy Selection

Tactic cost/benefit vectors

<table>
<thead>
<tr>
<th>Tactic</th>
<th>Response Time (R)</th>
<th>Malicious Clients (M)</th>
<th>Cost (C)</th>
<th>User Annoyance (A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>enlistServers</td>
<td>-1000</td>
<td>0</td>
<td>=</td>
<td>+1.0</td>
</tr>
<tr>
<td>lowerFidelity</td>
<td>-500</td>
<td>0</td>
<td>=</td>
<td>-0.1</td>
</tr>
<tr>
<td>addCaptcha</td>
<td>-250</td>
<td>-90</td>
<td>=</td>
<td>+0.5</td>
</tr>
<tr>
<td>forceReauthentication</td>
<td>-250</td>
<td>-70</td>
<td>=</td>
<td>+50</td>
</tr>
<tr>
<td>blackholeAttacker</td>
<td>-1000</td>
<td>-100</td>
<td>=</td>
<td>+50</td>
</tr>
<tr>
<td>throttleSuspicious</td>
<td>-500</td>
<td>0</td>
<td>=</td>
<td>+25</td>
</tr>
</tbody>
</table>

Utility functions

<table>
<thead>
<tr>
<th>( U_R )</th>
<th>( U_M )</th>
<th>( U_C )</th>
<th>( U_A )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 : 1.00</td>
<td>0 : 1.00</td>
<td>0 : 1.00</td>
<td>0 : 1.00</td>
</tr>
<tr>
<td>100 : 1.00</td>
<td>5 : 1.00</td>
<td>1 : 0.90</td>
<td>100 : 0.00</td>
</tr>
<tr>
<td>200 : 0.99</td>
<td>20 : 0.80</td>
<td>2 : 0.30</td>
<td></td>
</tr>
<tr>
<td>500 : 0.90</td>
<td>50 : 0.40</td>
<td>3 : 0.10</td>
<td></td>
</tr>
<tr>
<td>1000 : 0.75</td>
<td>70 : 0.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1500 : 0.50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000 : 0.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4000 : 0.00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Utility preferences

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Priority</th>
<th>( \omega_{U_R} )</th>
<th>( \omega_{U_M} )</th>
<th>( \omega_{U_C} )</th>
<th>( \omega_{U_A} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Minimizing number of malicious clients.</td>
<td>0.15</td>
<td>0.6</td>
<td>0.1</td>
<td>0.15</td>
</tr>
<tr>
<td>2</td>
<td>Optimizing good client experience.</td>
<td>0.3</td>
<td>0.3</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>3</td>
<td>Keeping cost within budget.</td>
<td>0.2</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Current state Aggregate impact Expected state
[1500,90,2,0] +[-250,80,0,25,50] = [1250,10,2.25,50]

Expected utility

\[
[U_R(1250), U_M(10), U_C(2.25), U_A(50)] = [0.625, 0.933, 0.25, 0.5]
0.625*0.3+0.933*0.3+0.25*0.1+0.5*0.3=0.6425
\]
Validating the Strategy Space

- Given an adaptation model:
  - Will the adaptation manager make reasonable strategy selections in all circumstances?
  - What will be the effect of those selections?

- Use probabilistic model checking to analyze properties of the adaptation model
  - Enables **exhaustive analysis of state space**
    - Quantitative properties
  - Translate adaptation models into PRISM specifications
    - Discrete-Time Markov Chains extended with rewards
  - Use reward–based probabilistic (PRCTL) properties to analyze
    - Strategy selections
    - Strategy impact on utility
Formal Model – Tactics and Strategies

- Target system encodes
  - System state
  - Tactic impact

- Adaptation strategies mirror Stitch strategy trees for the execution of tactics

```
Target System

```

```
module target_system
active_servers : [0..MAX_SERVERS] init init_active_servers;
cost : [0..MAX COST] init init_cost;
rt : [0..MAX RT] init init_rt; // Avg. Response time
mc : [0..100] init init_mc; // Malicious clients
ua : [0..100] init init ua; // User annoyance
lb ce : bool init init Tb ce; // Captcha enabled in LBs?

[addCaptcha] (lb ce) -> 1: (rt'=ac_f rt)
& (mc'=ac_f mc) & (cost'=ac_f cost) & (ua'=ac_f ua)
& (lb ce =true);
...
endmodule
```

```
Adaptation Strategies

```

```
module Challenge
node : [0..2] init 0;
leaf : bool init false;
end : bool init false;

[addCaptcha] (node=0) & (cNotChallenging) -> 1: (node'=1)
& (leaf'=true);
[forceReauthentication] (node=0) & (1cNotChallenging) -> 1: (node'=2)
& (leaf'=true);
[] (leaf) -> 1: (end'=true);
endmodule
```

```
Formal Model – Utility Profile

- Utility profile encodes utility functions and preferences as reward structures
  - Rewards incorporated to states corresponding to leaf nodes in model

Utility functions for DoS

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Priority</th>
<th>$w_{U_{1p}}$</th>
<th>$w_{U_{1m}}$</th>
<th>$w_{U_{1c}}$</th>
<th>$w_{U_{1k}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Minimizing number of malicious clients</td>
<td>0.15</td>
<td>0.6</td>
<td>0.1</td>
<td>0.15</td>
</tr>
<tr>
<td>2</td>
<td>Optimizing good client experience</td>
<td>0.3</td>
<td>0.3</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>3</td>
<td>Keeping cost within budget</td>
<td>0.2</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
</tr>
</tbody>
</table>

DoS utility profile encoding

```
formula uM = (mc >= 0 & mc <= 5 ? 1:0) + (mc > 5 & mc <= 20 ? 1 + ((0.80 - 1) * (mc - 5) / (20 - 5)) : 0) + (mc > 20 & mc <= 50 ? 0.80 + (0.40 - 0.80) * (mc - 20) / (50 - 20) : 0) + (mc > 50 & mc <= 70 ? 0.40 + (0.00 - 0.40) * (mc - 50) / (70 - 50) : 0) + (mc > 70 ? 0 : 0);
```

Rewards

```
rewards "rGU" // Global Utility
leaf & scenario=1 : 0.15 * uR + 0.6 * uM + 0.1 * uC + 0.15 * uA;
endrewards
```
Strategy Selection Analysis

- Based on quantifying expected utility after strategy execution
- Different preferences result in different strategy selections
- Choices are consistent

Minimize malicious clients
Scenario 1 ($w_{UH}=0.15, w_{Ud}=0.6, w_{UC}=0.1, w_{UA}=0.15$)

Optimize good client experience
Scenario 2 ($w_{UH}=0.3, w_{Ud}=0.3, w_{UC}=0.1, w_{UA}=0.3$)
Strategy Impact Analysis

- Quantify expected selected strategy impact on utility
  - $\Delta U = \text{Expected utility} - \text{Current utility}$

- No states show negative $\Delta U$
- Similar utility improvement across scenarios
  - Independent of strategy selections
Conclusions and Future Work

- **Principled approach to self-protection**
  - Compose existing mitigation tactics into strategies
  - Formally reason about strategy selection and impact
    - Security in the context of other business properties

- **Future work**
  - Extended validation
    - Further adaptation steps ahead
    - Additional properties
  - Proactive adaptation approaches (e.g., Moving target)