# *Architecture-Based Self-Protection: Composing and Reasoning about Denial-of-Service Mitigations*

## HotSoS 2014

Bradley Schmerl+, Javier Cámara+, Jeffrey Gennari*, David Garlan+, Paulo Casanova+, Gabriel A. Moreno*, Thomas J. Glazier+, Jeffrey M. Barnes+

Institute for Software Research+ and Software Engineering Institute*

**Carnegie Mellon University**

April 8, 2014

Carnegie Mellon

isr institute for SOFTWARE RESEARCH

# 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

* Yuan, E., Malek, S., Schmerl, B., Garlan, D., and Gennari, J. **Architecture-based self-protecting software systems**. In *Proceedings of the 9th International ACM Sigsoft Conference on the Quality of Software Architectures (QoSA 2013)*.

# In this Talk

- Formal reasoning about the composition of mitigation approaches

  - **W*hen* 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

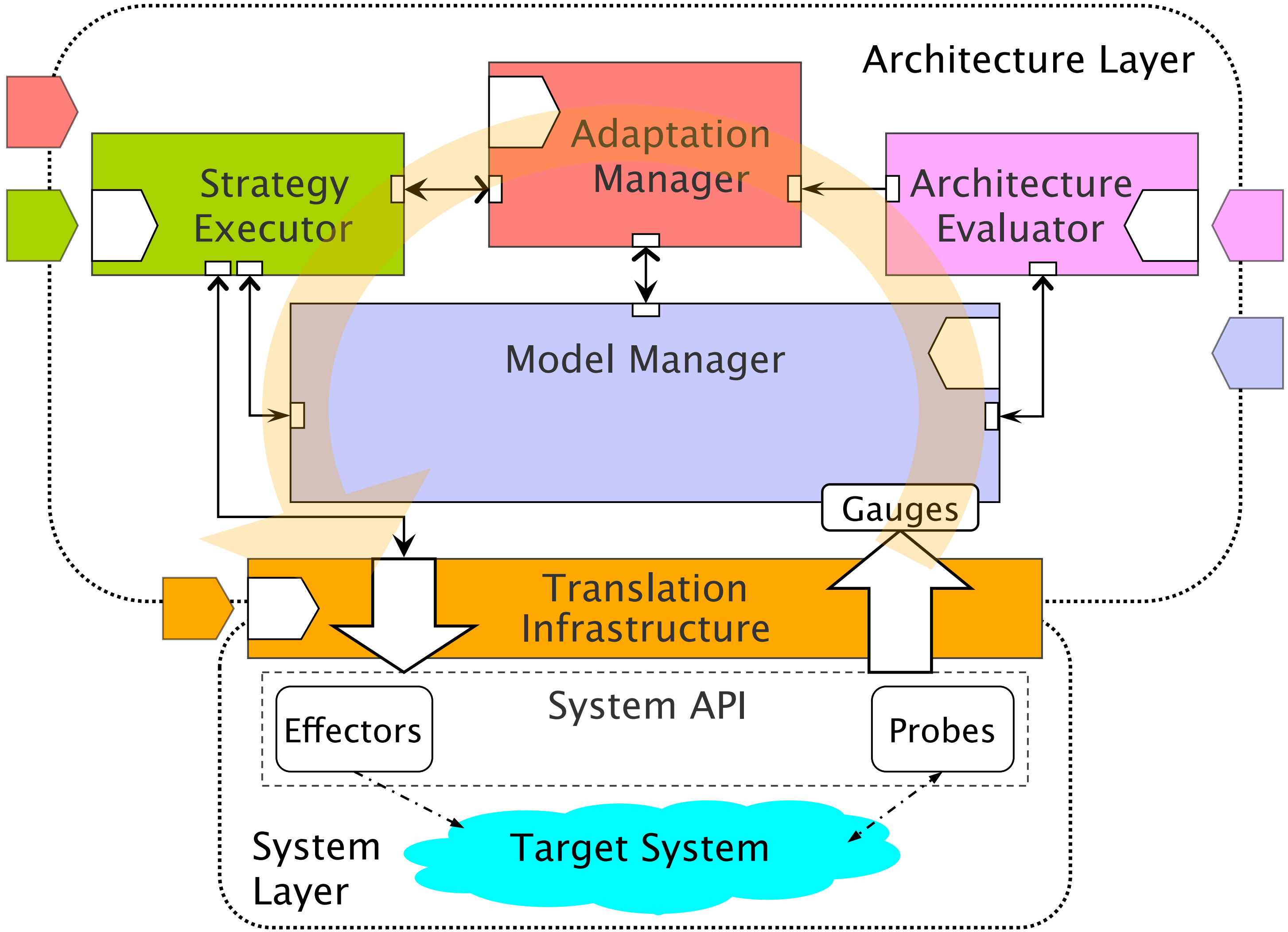
- **Architecture-based self-adaptation in Rainbow**
- **Example:** Denial of Service in Znn
- **Architecture-based self-protection in Rainbow**
- **Validating the strategy space**
  - Strategy selection analysis
  - Strategy impact analysis
- **Conclusions and future work**

# 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

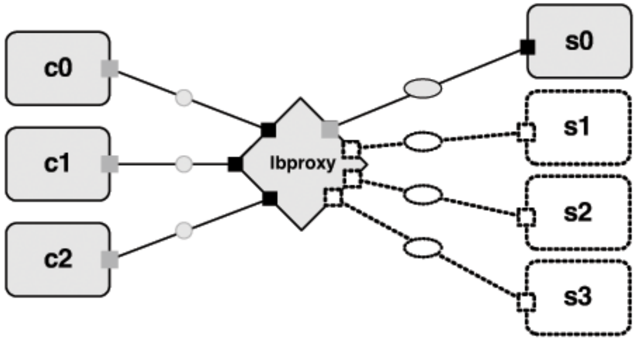
- **Control-system model:** Selection of next action in a strategy depends on observed effects of previous action
- **Value system:** Utility-based selection of best strategy allows context-sensitive adaptation
- **Asynchrony:** Explicit timing delays capture "settling time"
- **Uncertainty:** effect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

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

institute for SOFTWARE RESEARCH

# 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

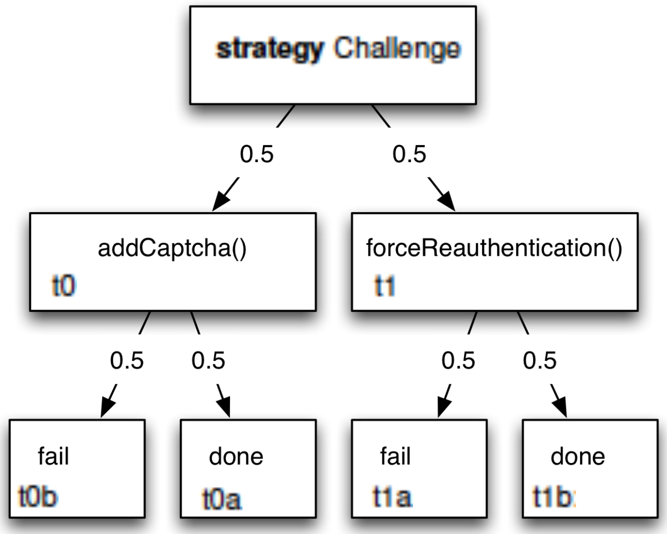
| Tactic | Description |
|---|---|
| Add capacity: | Activate additional servers to distribute the workload |
| Blackhole | Blacklists clients, requests are dropped |
| Reduce service | Reduce content fidelity level (e.g., text vs. images) |
| Throttle | Limits the rate of requests accepted |
| Captcha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

| Strategy | Description |
|---|---|
| Outgun/Absorb | Combines Add capacity and Reduce service |
| Eliminate | Combines Blackholing and Throttling |
| Challenge | Combines Captcha and Reauthenticate |

# Tactics and Strategies

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

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




institute for SOFTWARE RESEARCH

# Strategy Selection

## Tactic cost/benefit vectors

| Tactic | Response Time (R) | | Malicious Clients (M) | | Cost (C) | | User Annoyance (A) | |
|---|---|---|---|---|---|---|---|---|
| | Δ Avg. Resp. Time (ms.) | $\Delta U_R$ | Δ Malicious Clients (%) | $\Delta U_M$ | Δ Operating Cost (usd/hr.) | $\Delta U_C$ | Δ User Annoyance (%) | $\Delta U_A$ |
| enlistServers | -1000 | ↑↑↑ | 0 | = | +1.0 | ↓↓↓ | 0 | = |
| lowerFidelity | -500 | ↑↑ | 0 | = | -0.1 | ↑ | 0 | = |
| addCaptcha | -250 | ↑ | -90 | ↑↑↑ | +0.5 | ↓↓ | +50 | ↓↓ |
| forceReauthentication | -250 | ↑ | -70 | ↑↑ | 0 | = | +50 | ↓↓ |
| blackholeAttacker | -1000 | ↑↑↑ | -100 | ↑↑↑ | 0 | = | +50 | ↓↓ |
| throttleSuspicious | -500 | ↑↑ | 0 | = | 0 | = | +25 | ↓ |

## Utility functions

| $U_R$ | $U_M$ | $U_C$ | $U_A$ |
|---|---|---|---|
| 0 : 1.00 | 0 : 1.00 | 0 : 1.00 | 0 : 1.00 |
| 100 : 1.00 | 5 : 1.00 | 1 : 0.90 | 100 : 0.00 |
| 200 : 0.99 | 20 : 0.80 | 2 : 0.30 | |
| 500 : 0.90 | 50 : 0.40 | 3 : 0.10 | |
| 1000 : 0.75 | 70 : 0.00 | | |
| 1500 : 0.50 | | | |
| 2000 : 0.25 | | | |
| 4000 : 0.00 | | | |

Current state  Aggregate impact  Expected state

[1500,90,2,0] +[−250,80,0.25,50] =[1250,10,2.25,50]

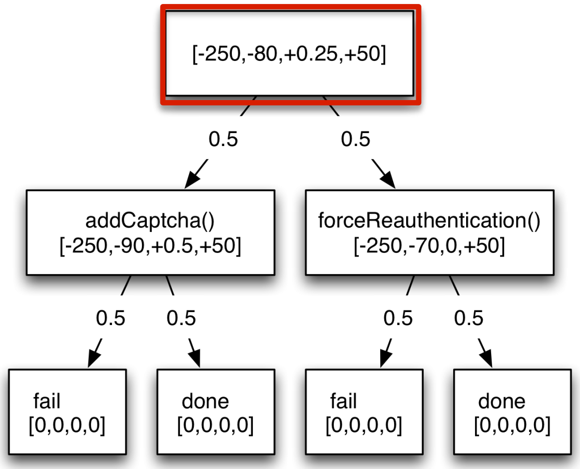
Expected utility

$[U_R(1250), U_M(10), U_C(2.25), U_A(50)] = [0.625, 0.933, 0.25, 0.5]$

$0.625*0.3+0.933*0.3+0.25*0.1+0.5*0.3=0.6425$

## Utility preferences

| Scenario | Priority | $w_{U_R}$ | $w_{U_M}$ | $w_{U_C}$ | $w_{U_A}$ |
|---|---|---|---|---|---|
| 1 | Minimizing number of malicious clients. | 0.15 | 0.6 | 0.1 | 0.15 |
| 2 | Optimizing good client experience. | 0.3 | 0.3 | 0.1 | 0.3 |
| 3 | Keeping cost within budget. | 0.2 | 0.2 | 0.4 | 0.2 |



# 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

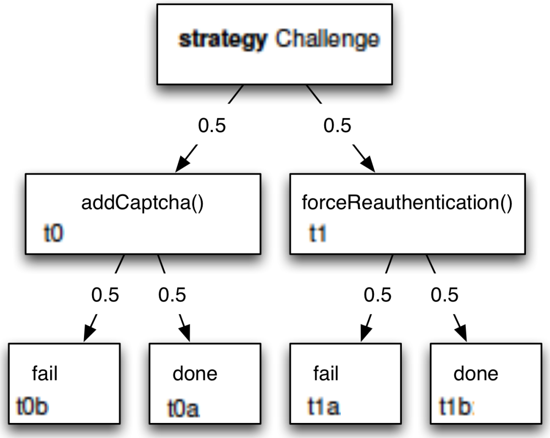
# Formal Model – Tactics and Strategies

- Target system encodes
  - System state
  - Tactic impact
- Adaptation strategies mirror Stitch strategy trees for the execution of tactics

Target System

```
formula ac_f_rt=rt-250>=0?(rt-250<=MAX_RT?rt-250:MAX_RT):0;
...
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_lb_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)
                          & (!cNotChallenging) -> 1: (node'=2)
                          & (leaf'=true);
  [] (leaf) -> 1: (end'=true);
endmodule
```

institute for SOFTWARE RESEARCH

# 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

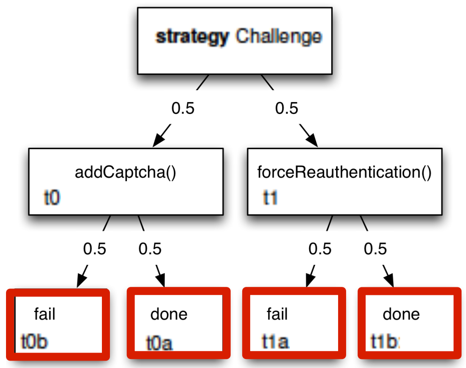
| $U_R$ | $U_M$ | $U_C$ | $U_A$ |
|---|---|---|---|
| 0 : 1.00 | 0 : 1.00 | 0 : 1.00 | 0 : 1.00 |
| 100 : 1.00 | 5 : 1.00 | 1 : 0.90 | 100 : 0.00 |
| 200 : 0.99 | 20 : 0.80 | 2 : 0.30 | |
| 500 : 0.90 | 50 : 0.40 | 3 : 0.10 | |
| 1000 : 0.75 | 70 : 0.00 | | |
| 1500 : 0.50 | | | |
| 2000 : 0.25 | | | |
| 4000 : 0.00 | | | |

Utility preferences for DoS

| Scenario | Priority | $w_{U_R}$ | $w_{U_M}$ | $w_{U_C}$ | $w_{U_A}$ |
|---|---|---|---|---|---|
| 1 | Minimizing number of malicious clients. | 0.15 | 0.6 | 0.1 | 0.15 |
| 2 | Optimizing good client experience. | 0.3 | 0.3 | 0.1 | 0.3 |
| 3 | Keeping cost within budget. | 0.2 | 0.2 | 0.4 | 0.2 |

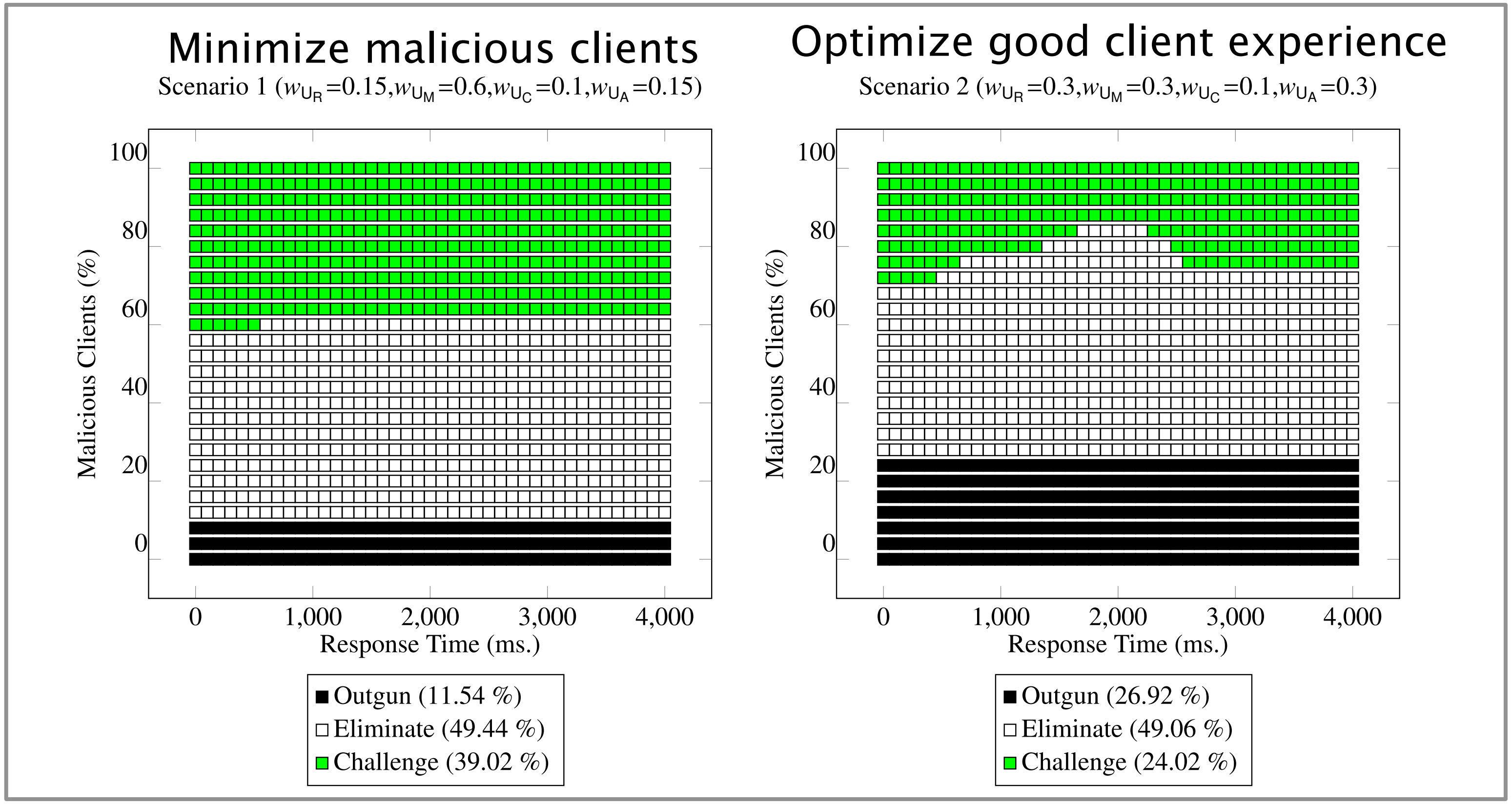
DoS utility profile encoding

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



# Strategy Selection Analysis

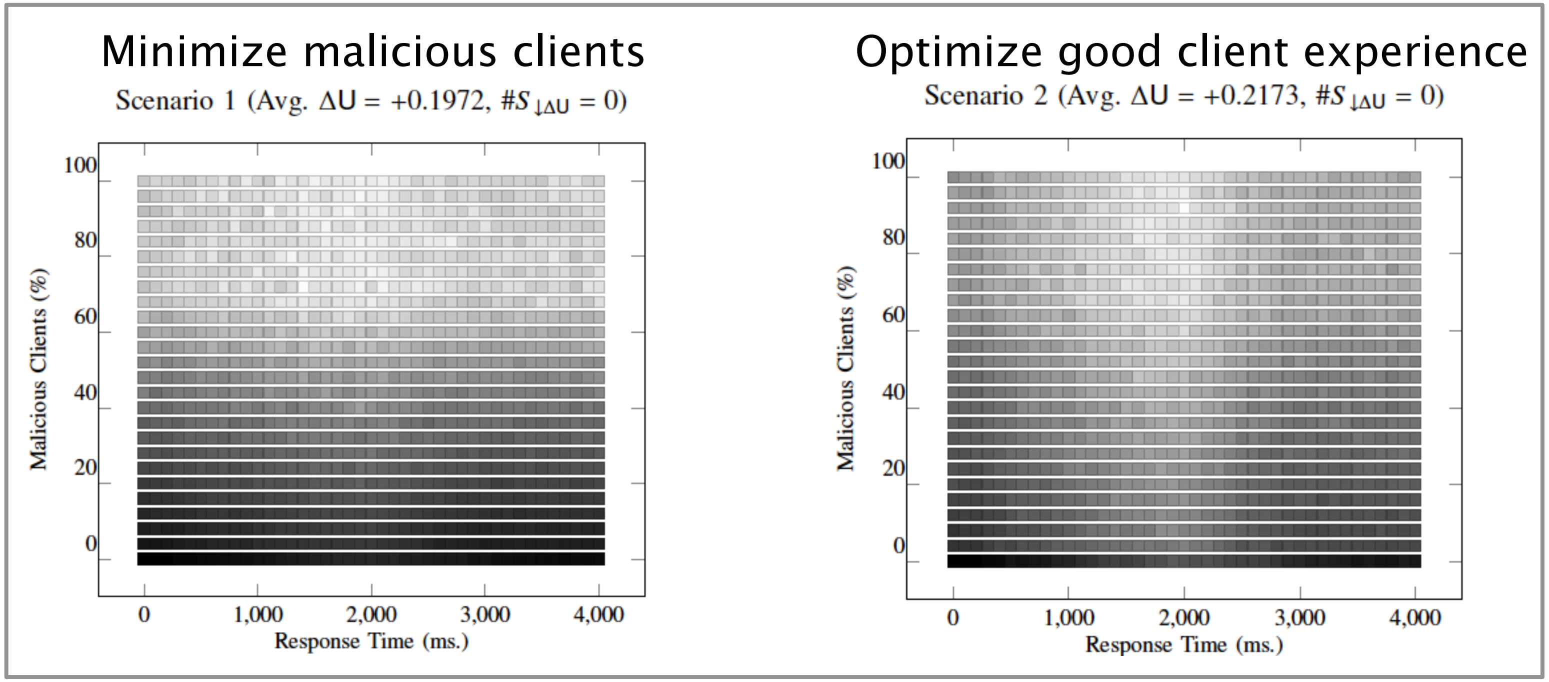
- Based on quantifying expected utility after strategy execution
- Different preferences result in different strategy selections
- Choices are consistent




institute for SOFTWARE RESEARCH

# Strategy Impact Analysis

- Quantify expected selected strategy impact on utility
  - ΔU = Expected utility – Current utility




- No states show negative ΔU
- 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)