Specification-driven Moving Target Defense Synthesis

Md Mazharul Islam, Qi Duan and Ehab Al-Shaer
University of North Carolina Charlotte
Charlotte, North Carolina, USA
{mislam7,qduan,ealshaer}@uncc.edu

Motivation

- Cyber defense techniques are mostly
  - **Non-adaptive:** take a long time to detect and respond against adversary
  - **Rigid:** do not provide agility in mitigating threat **proactively**.
- Static and predictable behavior of cyber systems from the attackers' view creates a fundamental design vulnerability.
- Cyber agility enables cyber systems to defend proactively against sophisticated attacks by
  - **dynamically changing** the system configuration parameters: **mutable parameters**
  - **deceive adversaries** from reaching their **goals**
  - **disrupt the attack plans** by forcing them to change their **adversarial behaviors**
  - **deterring adversaries** through prohibitively increasing the **cost for attacks**

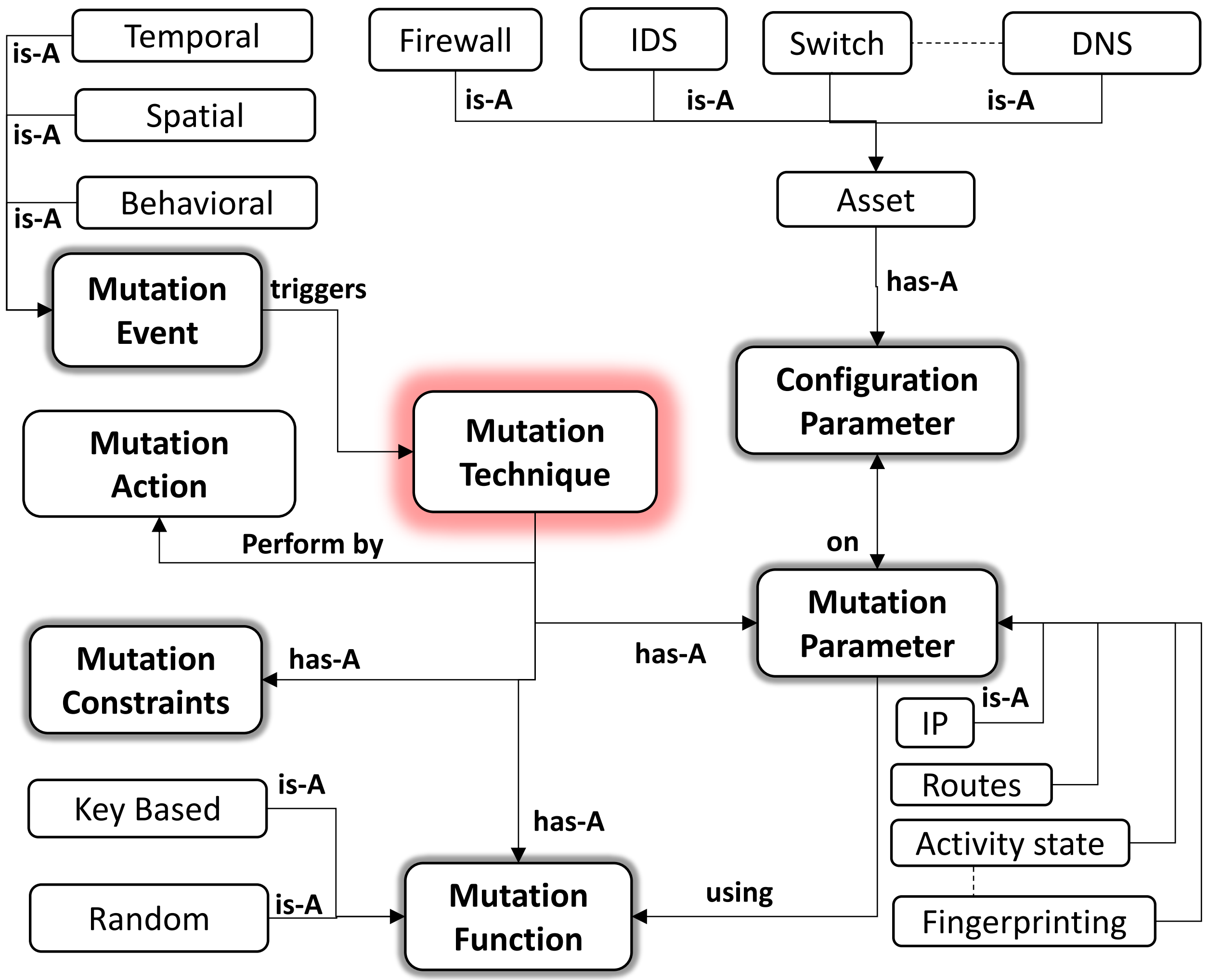
Problem Definition

- Developing cyber agility such as **moving target defense (MTD)** techniques that are provable safe is a **highly complex task**
- Requires significant **time and expertise** in implementation and management
- Requires low level configuration changes:
  - Dynamically
  - Periodically
- Which can break:
  - The mission integrity
  - The goal of the defense
  - Reachability of the network
- Because of
  - Misconfiguration
  - Conflicts with existing policies

Our Approach

- Our goal is to address this challenge by providing **a framework** for automating the creation of configuration-based moving target techniques **rapidly and safely**.
- We developed MTDSynth: a cyber agility framework allowing MTD developers for creating MTD control programs using a **high-level cyber agility policy language** (HAPL)
- MTDSynth ensures the safe orchestrion and deployment of MTD policies by the followings:
  - *Mutation triggers:* time-based or event-based using user-defined network sensors
  - *MTD mutable parameters:* dynamically changed based on the trigger
  - *Configuration parameters:* dependent on the mutable parameters
  - *Mutation functions constraints:* dictates the methodology to compute and optimize the selection of new mutation value
  - *Mutation attributes:* define the mutation scope or domain
- MTDSynth also provides
  - **A policy refinement engine** to synthesize the control program using Software-defined networking (SDN)
  - **A translator** to verify the MTD control programs that satisfy the constraints defined in the agility policy specification.
- MTDSynth provides an **open programming environmen**t to develop **rapidly and safely** sense-making and decision making MTD actions to enable cyber agility for dynamic cyber defense.

Cyber Agility Policy Ontology for MTDSynth



High-level cyber agility policy language (HAPL)

Agility Rule **Π** ::= N : E → Λ
MTD Name **N** ::= **STRING**
Mutation Event **E** ::= Δ | α
Time Interval **Δ** ::= NUMBER
Sensor Alert **α** ::= *isHostScanning()* | *isLinkFlooding()*
| *isBotDetected()* | *checkUDPICMPPRate()*
| *getAvailableBandwidth()* | *checkNewComers()*
| *checkElephantTCP()* | *getRouteLength()*
| *getCriticalLink()* | *getRouteRisk()*
| *getFlowStatistics()* | *getAllFlowRules()*
| *getFlowRate()*
Agility Spec. **Λ** ::= **MUTATE** p id **OF** {attr } **USING** f
ON g **BY** m **WHILE** c
Mutation Param. **p** ::= ROUTE | IP | STATE
IP Attr. **ipattr** ::= id .I P → ⟨l ist of IPAddress⟩ | h

Mutation Func. **f** ::= random(η, η) | key-based
Configuration **g** ::= r | r, g
Resource **r** ::= DNS-entry | switch-table | s
Mutation Action **m** ::= *ipMutate* | *pathMutate* | *reDirect*
| *spatalMutation* | *createShadow* | *migrateService*
Constraint Spec **c** ::= β | β ; c
Agility Const. **β** ::= α | γ | δ | δ op v
Mutation Const. **γ** ::= id_{t+1} op η | id_{t+1} op id_t
| $attr_{t+1}$ op η $attr_{t+1}$ op $attr_t$
Network Func. **δ** ::= *includeSwitch()* | *excludeSwitch()*
| *overlap()* | *canReach()* | *getAllPaths()*
| *getShortestPath()* | *getMinDetectionProb()*
| *getAttackUncertainity()*
Operator **op** ::= > | < | ≤ | ≥ | = | , | ∩ | ∪
| ∀ | ∃ | + | − | × | /

MTD Policy Examples: Route Mutation

Route Mutation:

isLinkFlooding(l, 0.2) $\rightarrow$

MUTATE route R of {R.src $\rightarrow$ IP_1, R.dst $\rightarrow$ IP_2}

USING random(1..N) **ON** *switch-table* **BY** *pathMutate*

WHILE

$(R_t \cap R_{t+1}) / R_t \geq 0.7$;

includeSwitch(R_{t+1}, [s_2]) == **TRUE**;

excludeSwitch R_{t+1}, [s_6]) == **TRUE**;

getRouteLength(R_{t+1}) ≤ 5;

getAvailableBandwidth(R_{t+1}) > *getFlowRate*(IP_1, IP_2) $\times 1.2$;

getRouteRisk(R_{t+1}) ≤ 0.25

MTDSynth API: Route/Path Mutation

Defense Actions	Parameters	Descriptions	
pathMutate()	<**src**>	List of source Host IPs, e.g. <192.168.10.20/32, 192.168.10.75/32, 192.168.55.99/32, …> etc.	
	<**dst**>	List of destination Host IPs, e.g. <192.168.10.20/32, 192.168.10.75/32, …> etc. ***Note:*** (src[i], dst[i]) must be the end hosts of a complete path, where src[i] and dst[i] is the *I*'th element of each list.	
	pathProfile Example: pathProfile, P = (v,n,B,R) v = overlap n = number of links in a path B = Maximum bandwidth R = Maximum Risk threshold	**overlap**	**0**: No overlap.
			percentage: How much overlap is acceptable.
			excludeSpecificLinks: Exclude any specific link form the path
			includeSpecificLinks: Include any specific link form the path
		maxPathLength: User can provide the maximum path length which can be used as new path.	
		availableBandwith: User can provide the maximum bandwidth of each link in a path.	
		maxRisk = R, where $(1 - (1-p)^n) < R$ (p: prob that a link is under attack)	For Each link of any path containing *n* links, the probability of that link is under attack is *p.*
	pattern	**-1**: Deactivate path mutation.	
		0: Immediate mutation (single mutation)	
		x: Temporal mutation. Mutate path after every *x* seconds (periodic /continues mutation)	
		Conditional (as long as): Mutate path after every *x* seconds until a condition.	

MTD Policy Examples: Spatial Mutation

Spatial IP Mutation:
isHostScanning(100, 5) →
MUTATE IP P of {P.IP → [h_1, h_2 ,…, h_m]}
USING random(1..N) **ON** *DNS-entry* **BY** *spatialMutation*
WHILE
$m \times (m-1)/N \leq 0.1$;
$\forall_{i,\, j \in N}\, P_t \,.\, h_i \neq P_{t+1} \,.\, h_i$

MTDSynth API: Spatial Mutation

MTD Actions	Parameters	Descriptions
spatialMutation()	***<h>***	List of ***h*** mutable host.
	N	Number of total host in the network.
	<unused_range>	Unused IP address list. (Also can be provided with start address and range limit.)
	< m_i >	Mutable address per host = $(n-1)*m_i$
	when	**-1**: Deactivate Special IP mutation.
		0: One time mutation.
		x: Time based mutation. Mutate IP after *x* seconds.
		other-mutation: For future use.
	how	**Uniform Distribution, Random Distribution**: Select mutable address list for each mutable host from ***<unused_range>***

MTD Policy Examples: IP Mutation

Temporal IP Mutation:
timeInterval = 5s →
MUTATE IP P of {P.IP → [h_1, h_2 ,…, h_n]}
USING random(1..N) **ON** *DNS-entry* **BY** *ipMutate*
WHILE

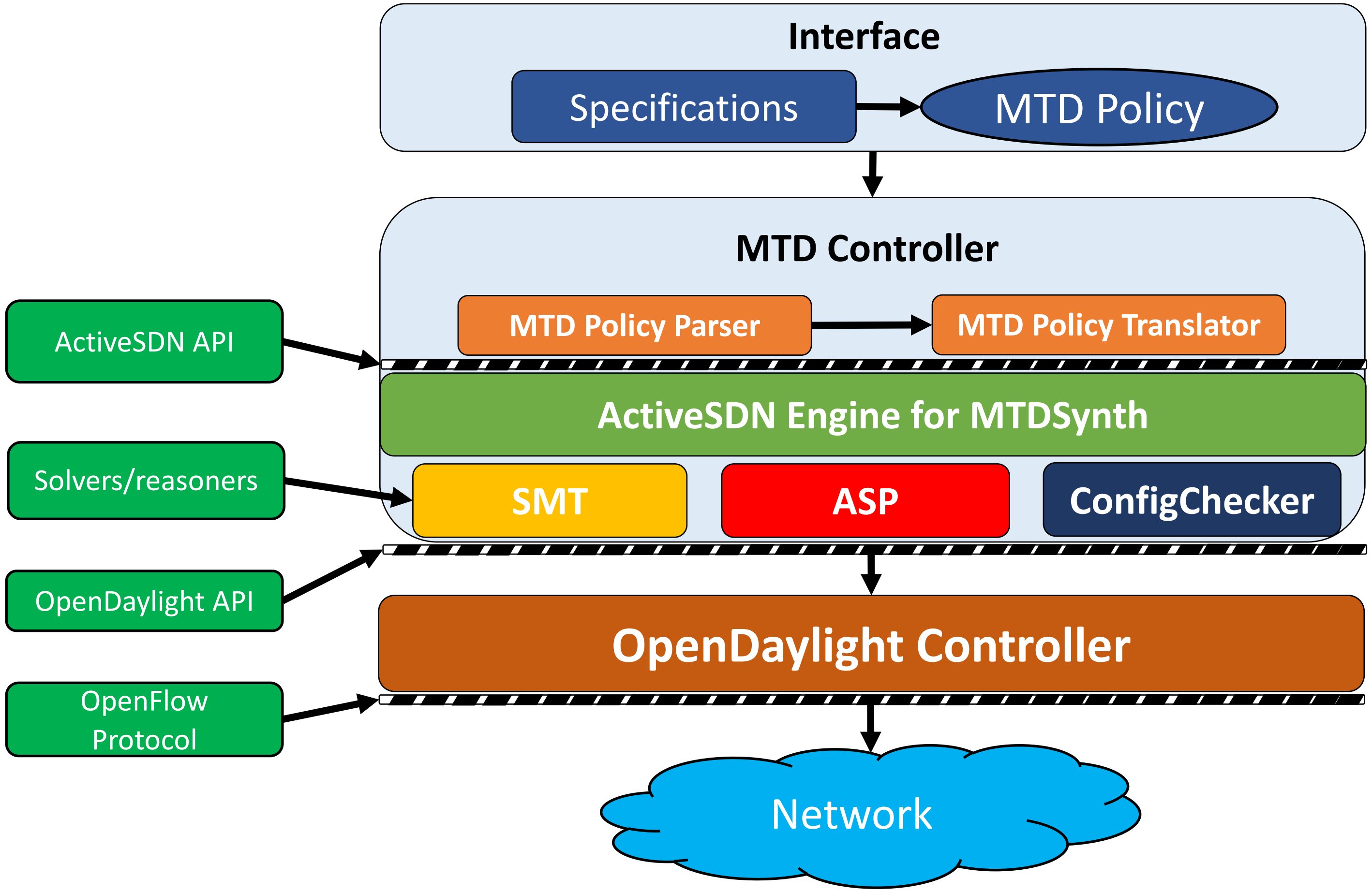
$\forall_{i,\, j \in (1,\, N)}\ P_{t+1} . h_i \neq P_{t+1} . h_j;$

$\forall_{i,\, j \in N}\ P_t . h_i \neq P_{t+1} . h_i$

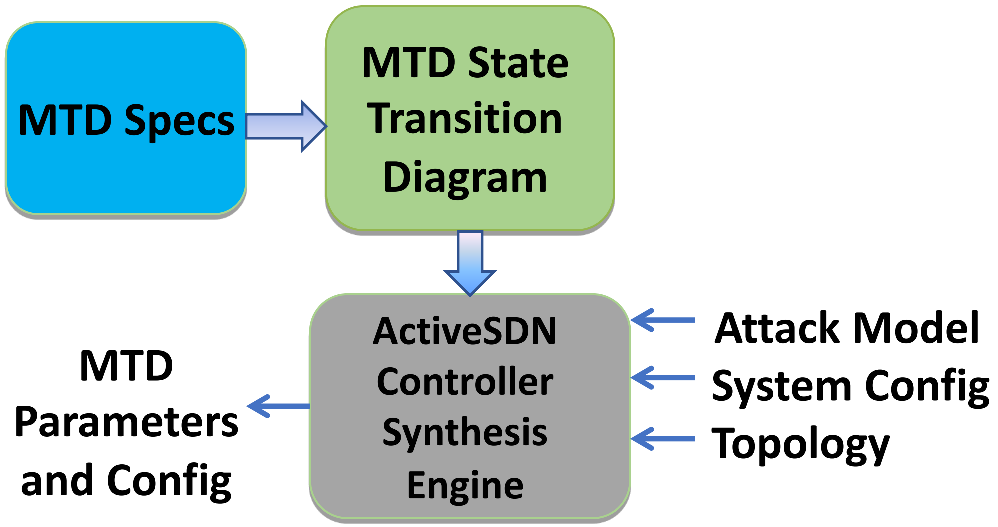
MTDSynth API: IP Mutation

Defense Actions	Parameters	Descriptions
ipMutate()	<**rIP**>	List of initial real IPs, e.g. <192.168.10.20/32, 192.168.10.21/32, 192.168.55.99/32,…> etc.
	when	**-1**: Deactivate IP mutation.
		0: One time mutation.
		x: Time based mutation. Mutate IP after *x* seconds.
		other-mutation: For future use.
	how	**specific vIP**: User defined virtual IP, e.g. 192.168.60.99
		randomFunction(): A function which will provide random vIPs in a specific time window *x.*

MTD Controller Synthesis using MTDSynth



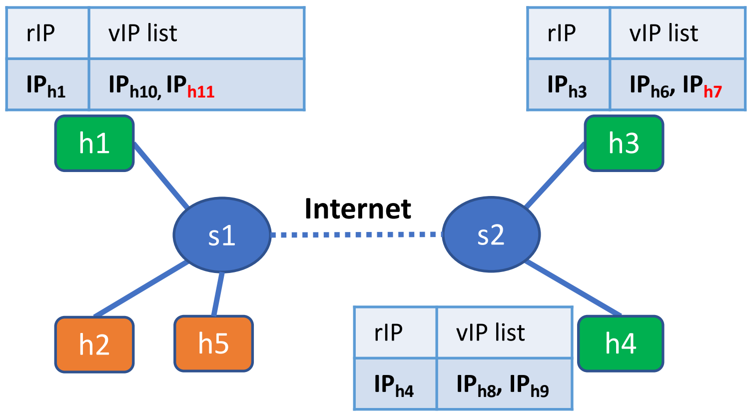
MTD Controller Synthesis Controller



The task of the MTDSynth Synthesizer and control system is to generate a sequence of configuration control signals that, by construction, ensures that the system satisfies the model requirements for the MTD techniques, given the

- Specifications for MTD techniques (parameters, actions, and constraints expressed in Linear Temporal Logic (LTL)),
- Environment specifications (attack model, topology and system configurations),

Case Study: Spatial Mutation



Netwrok

h1→ (h3, **IP_{h7}**), (h4, **IP_{h8}**)
h3→ (h4, **IP_{h9}**), (h1, **IP_{h11}**)
h4→ (h3, **IP_{h6}**), (h1, **IP_{h10}**)

DNS-entry

Flow	s1	Internet	s2	Flow
$IP_{h1} \rightarrow IP_{h7}$	❶ src=IP_{h1},dst=IP_{h7};set_dst:IP_{h3}	$IP_{h1} \rightarrow IP_{h3}$	❷ src=IP_{h1},dst=IP_{h3};set_src:IP_{h11}	$IP_{h11} \rightarrow IP_{h3}$
$IP_{h1} \leftarrow IP_{h7}$	❸ src=IP_{h3},dst=IP_{h1};set_src:IP_{h7}	$IP_{h1} \leftarrow IP_{h3}$	❹ src=IP_{h3},dst=IP_{h11};set_dst:IP_{h1}	$IP_{h11} \leftarrow IP_{h3}$

Flow Entry in edge switches

Spatial Mutation: Flow Rules in SDN Switch Tables

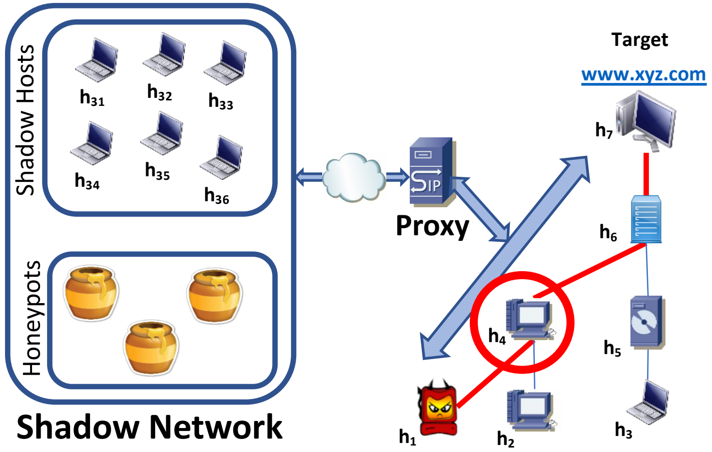
```
Every 3.0s: sudo ovs-ofctl dump-flows -OOpenFlow13 s1                                                    Wed Jul 10 12:26:18 2019

OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x6e, duration=1220.462s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.3,nw_dst=10.0.0.1 actions=set_field:10.0.0.7->ip_src,output:1,set_field:0->ip_dscp
 cookie=0x6b, duration=1220.480s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.1,nw_dst=10.0.0.7 actions=set_field:10.0.0.3->ip_dst,set_field:f2:5c:62:05:ca:c9->eth_dst,output:3,set_field:0->ip_dscp
 cookie=0x5b, duration=1220.597s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.2,nw_dst=10.0.0.5 actions=set_field:10.0.0.3->ip_dst,set_field:f2:5c:62:05:ca:c9->eth_dst,output:3,set_field:0->ip_dscp
 cookie=0x5e, duration=1220.585s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.3,nw_dst=10.0.0.2 actions=set_field:10.0.0.5->ip_src,output:2,set_field:0->ip_dscp
 cookie=0x5f, duration=1220.561s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.2,nw_dst=10.0.0.9 actions=set_field:10.0.0.4->ip_dst,set_field:4e:8c:33:4e:5c:42->eth_dst,output:3,set_field:0->ip_dscp
 cookie=0x62, duration=1220.548s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.4,nw_dst=10.0.0.2 actions=set_field:10.0.0.9->ip_src,output:2,set_field:0->ip_dscp
 cookie=0x63, duration=1220.537s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.2,nw_dst=10.0.0.12 actions=set_field:10.0.0.1->ip_dst,set_field:e2:b3:16:8c:34:d2->eth_dst,output:1,set_field:0->ip_dscp
 cookie=0x64, duration=1220.534s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.1,nw_dst=10.0.0.2 actions=set_field:10.0.0.12->ip_src,output:2,set_field:0->ip_dscp
 cookie=0x72, duration=1220.439s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.4,nw_dst=10.0.0.1 actions=set_field:10.0.0.10->ip_src,output:1,set_field:0->ip_dscp
 cookie=0x6f, duration=1220.456s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.1,nw_dst=10.0.0.10 actions=set_field:10.0.0.4->ip_dst,set_field:4e:8c:33:4e:5c:42->eth_dst,output:3,set_field:0->ip_dscp
 cookie=0x0, duration=1258.203s, table=0, n_packets=279, n_bytes=23045, priority=150 actions=CONTROLLER:65535
 cookie=0x2b00000000000007, duration=1258.203s, table=0, n_packets=0, n_bytes=0, priority=1,arp actions=CONTROLLER:65535
```

```
Every 3.0s: sudo ovs-ofctl dump-flows -OOpenFlow13 s3                                                    Wed Jul 10 12:28:58 2019

OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x66, duration=1379.965s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.3,nw_dst=10.0.0.8 actions=set_field:10.0.0.6->ip_src,set_field:10.0.0.4->ip_dst,set_field:4e:8c:33:4e:5c:42->eth_dst,output:2
,set_field:0->ip_dscp
 cookie=0x65, duration=1379.975s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.4,nw_dst=10.0.0.6 actions=set_field:10.0.0.8->ip_src,set_field:10.0.0.3->ip_dst,set_field:f2:5c:62:05:ca:c9->eth_dst,output:1
,set_field:0->ip_dscp
 cookie=0x6d, duration=1379.915s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.3,nw_dst=10.0.0.11 actions=set_field:10.0.0.1->ip_dst,set_field:e2:b3:16:8c:34:d2->eth_dst,output:3,set_field:0->ip_dscp
 cookie=0x6c, duration=1379.920s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.1,nw_dst=10.0.0.3 actions=set_field:10.0.0.11->ip_src,output:1,set_field:0->ip_dscp
 cookie=0x5c, duration=1380.036s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.2,nw_dst=10.0.0.3 actions=output:1,set_field:0->ip_dscp
 cookie=0x5d, duration=1380.033s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.3,nw_dst=10.0.0.2 actions=output:3,set_field:0->ip_dscp
 cookie=0x60, duration=1379.998s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.2,nw_dst=10.0.0.4 actions=output:2,set_field:0->ip_dscp
 cookie=0x61, duration=1379.995s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.4,nw_dst=10.0.0.2 actions=output:3,set_field:0->ip_dscp
 cookie=0x71, duration=1379.887s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.4,nw_dst=10.0.0.13 actions=set_field:10.0.0.1->ip_dst,set_field:e2:b3:16:8c:34:d2->eth_dst,output:3,set_field:0->ip_dscp
 cookie=0x70, duration=1379.890s, table=0, n_packets=0, n_bytes=0, priority=400,ip,nw_src=10.0.0.1,nw_dst=10.0.0.4 actions=set_field:10.0.0.13->ip_src,output:2,set_field:0->ip_dscp
 cookie=0x0, duration=1417.925s, table=0, n_packets=598, n_bytes=50328, priority=150 actions=CONTROLLER:65535
 cookie=0x2b00000000000005, duration=1417.687s, table=0, n_packets=0, n_bytes=0, priority=1,arp actions=CONTROLLER:65535
```

Case Study: Deception by MTDSynth

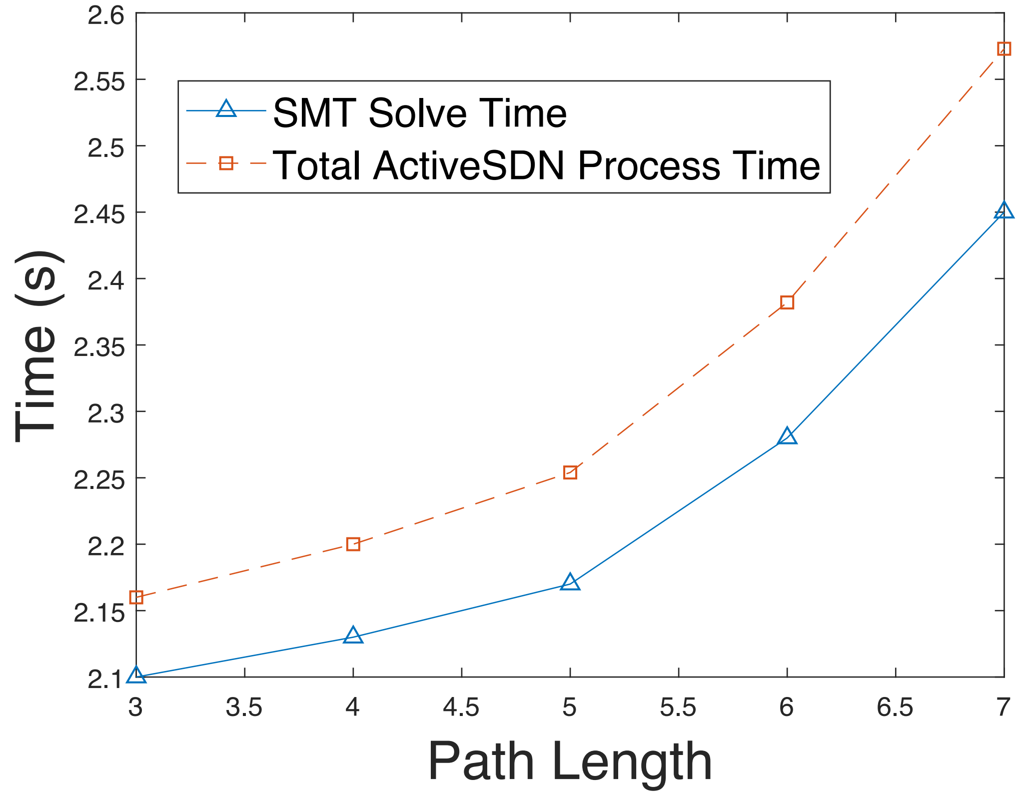


Deception by MTDSynth

Proxy
$(src=IP_{adversary}, dst=IP_{h4}) \rightarrow (src=IP_{proxy}, dst=IP_{honeypot})$
$(src=IP_{honeypot}, dst=IP_{proxy}) \rightarrow (src=IP_{h4}, dst=IP_{adversary})$

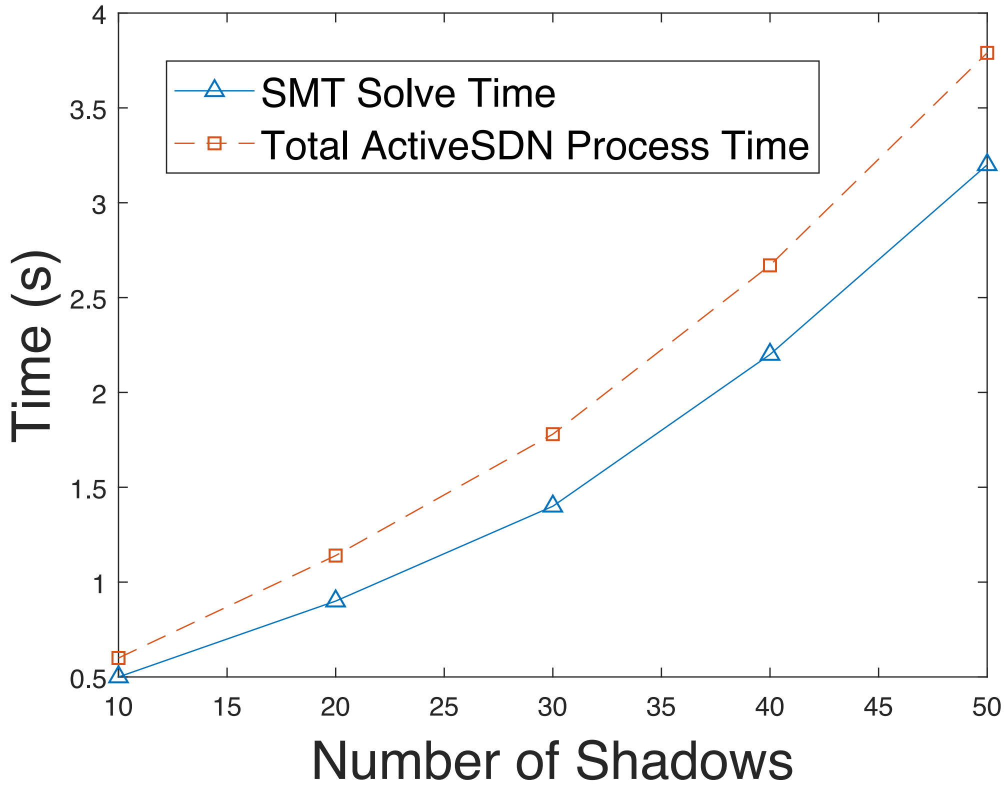
Flow rules in the proxy

Evaluation: ActiveSDN Overhead in Terms of Path Length (for RRM)



- **We compare the total processing delay and the SMT solve time for RRM in a network of 200 hosts with a flow of fixed source and destination, and different required path length.**
- **The ActiveSDN configuration delay is small (for example, about 0.07s for path length 4, and 0.13 s for length 7), which means the ActiveSDN overhead is acceptable.**

Evaluation: ActiveSDN Overhead in Terms of Number of Shadows



- **We compare the total processing delay and the SMT solve time of the shadow host planning in a network with 12 mutable hosts, with different number of shadow addresses.**
- **The difference between SMT solve time and total processing delay is small, which means the configuration delay is small (for example, about 0.24s for 20 shadows).**

MYDSynth API

MTD Action
ipMutate
pathMutate
spatialMutation
createShadow
reDirect
migrateService

Sensors Action
isHostScanning(th, t)
isLinkFlooding(l, th)
chekUDPICMPRate(f)
checkElephantTCP(<f>)
getFlowStatistics(f)
checkNewComers(<f>, t)
getCriticalLinks()
getAllFlowRules (s)
findNeighbors(s)
findPortID(l, r)
detectBot(sig)

Constraints
isIncludeSwitch(R_t, <s>)
excludeSwitch(R_t, <s>)
getRouteLength(R_t)
getAvailableBandWidth(R_t)
getFlowRate(s, d)
overlap(R_t, R_{t+1})
getRouteRisk(R_t)
canReach(s, d)
checkUniqueIP(<ip>)
checkNonRepeate (<ip1>, <ip2>)
checkSpatialCollision(<ip1>, <ip2>)
getMinDetectionProb(loc)
getAttackUncertainity(loc)
getAllPaths(s, d)
getShortestPath(s, d)