Intrusion detection system (IDS):

It is hardware or software system that automate the process of monitoring the events occurring in computer system or network analysing them for signs of security problems.

Intrusion Response System (IRS):

It is automatically responds to these attack (by attacker).

Previous work that for IDS/IRS:

- Model base method
- Subjective performance values

The challenges with above two approach is due to dynamic nature of network topologies and system configuration.

Early IRS include the static mapping between the known alerts and practical response, this have a problem in which the size of complexity of computer network.

Now a days work:

Now a days the response is shift towards a dynamic response approach based on modelling response selection as a multi objective optimization problem.

This approach involves the use of

- Attack graph (AG)
- Attack tree (AT)

RL Implementation:

- **Model Based:**
  Here the agent learns a model of environment and use this model to plan the action

  It gives optimal result, difficult in complex environment, less interactive with environment.

- **Model Free:**
  Here the agent directly learns a policy/values function without explicitly learning a model of the environment.
  It gives not optimal result, simple to implement, robust to environment, more interaction with environment.

The contribution of this research paper is:

- Practical implementation of model-free RL based IRS, using virtual network, and simulated approach.
- Measure the cost and benefits of applying a response in a live network environment in the form of reward function.
- How the RL agent to learn an effective response to multi stage attack scenarios, building on related work which experiment with the single attack step or simulate the compromised network state.

Previous work before model -free RL based IRS:

Before ML:

- ADEPTS: It introduce the concept of I-graph (Attack graph) to map possible attacker routes in network manually.

- NICE: It merges the IDS with alert based VM state, further it improves by attack graph.

- Shameli-Sendi's approach: It introduce the dynamic cost-benefits analysis parameter including the administrative approach, it is further updated using number of firewall rules in use.

After ML:

- Auto Selection of Security Service Function Chaining: It use the MDP with discrete state space, it applied the Q-learning to select the optimal security function chaining (SFC) for software defined network. And it reliance on human scored SFC performance.

- Zolotukhin: it focuses on action to single-step attack only like brute force attack, DNS Tunnelling, and Distributed Denial of Service (DDoS).

- Miehling: They use a belief system based on previous experience and probabilities of future state to incorporate the unknown attack state into the response policy.

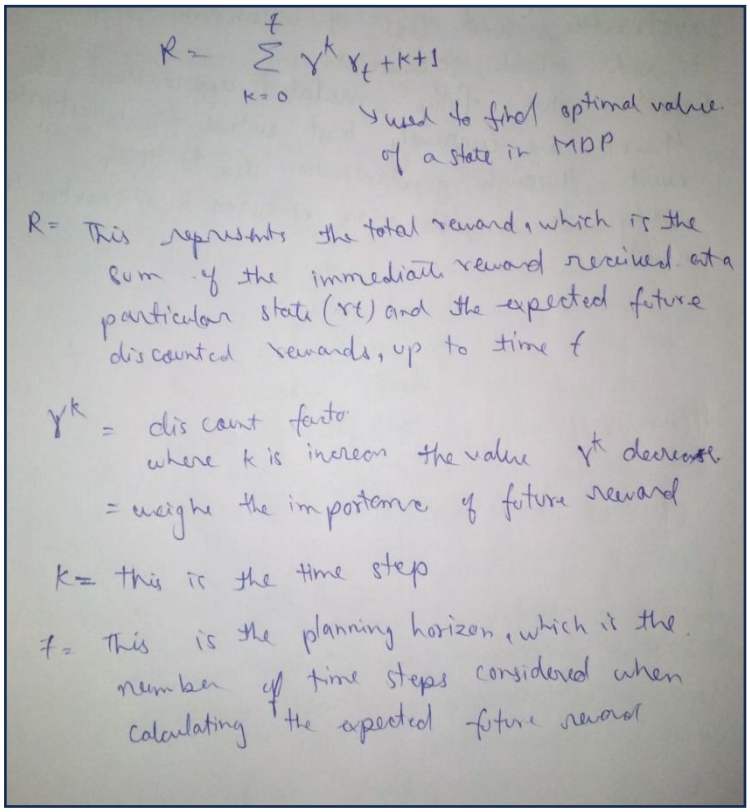
- Lannucci:

  It proposed an approach to address the state space explosion problem in IRS research by automatically.
  Most effective work is proposed by this that address the limitations of current model-based IRS by implementing a model free approach using Deep Q- learning.
  They develop a software simulation to replicate the system the IRS is designed to protect, enabling and training on simulated model before deployment on the live system

How reward and Value is achieves by agent in RL:

In RL the agent maximize the accumulated reward R gained over a number of state changes known as timestep.

The value and reward calculate using the Bellman equation.



Reward: RL agent needs feedback signals called rewards to know if their actions are good or bad. These rewards come from the environment after the agent taken an action.

Reward is two types:

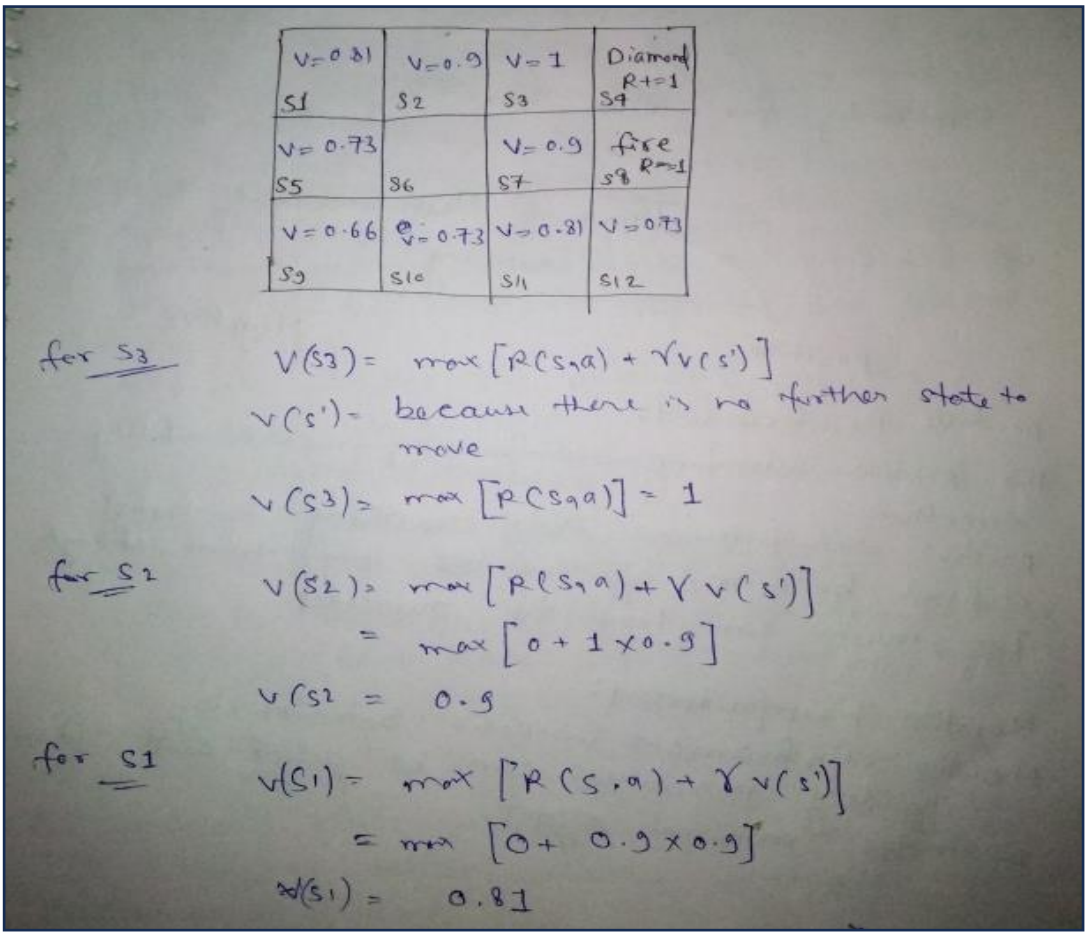
- Extrinsic: from the environment like getting closer to the finish line. Model free RL use this.
- Intrinsic: generated by the agent itself.

Value, Optimal value function, Optimal policy calculations:

- Action performed by the agent is referred to as "a"
- State occurred by performing the action is "s."
- The reward/feedback obtained for each good and bad action is "R."
- A discount factor is Gamma "γ."

The Bellman equation can be written as:

$$V(s) = \max [R(s,a) + \gamma V(s`)]$$



$$V^*(s) = max_a \{\sum_{s'} P_a(s) (R_a(s'|s, a) + \gamma V(s'))\}$$

$$\pi^*(s) = \arg max_a \{\sum_{s'} P_a(s) (R_a(s'|s, a) + \gamma V_i(s'))\}$$

The above two formula is Optimal Value function and Optimal policy function respectively:

For example total values are (if we start from S5 and destination is S4):

- V1 = S5 + S1 + S2 + S3 + S4 = 3.44
- V2 = S5 + S9 + S10 + S11 + S7 + S3 + S4 = 4.83
  Similarly other paths also exist that depends on environment.
- Here the total optimal value is V1
- Optimal Policy Function always get the optimal total value in which values sum is maximum.
- The Optimal Policy Function helps to find best action.

Model Free RL:

This modelled environment grows the memory and computationally resource need to maintain this model grows exponentially.

Why we use it:

- Real world complexity.
- Trial and error learning
- Finding the best action: in a vast environment the agent can't try every possibility. It needs to balance between the exploring new action and exploiting actions already known to be good.
- Hyperparameter for controls: Parameter like learning rate in Q-learning. low learning rate means prioritizing old knowledge (exploitation) while a high learning rate favour new experiences (exploration)

Model free RL algorithm:

- Proximal policy optimization (PPO):
  PPO limits how much the policy changes with each update. This uses a technique called clipping to prevent overly drastic changes.
  And it is reliable and easy to use
- Deep Q-Learning

Methodology of model-free approach to IRS:

why we apply model free IRS:

- Defender Dilemma: we can not predict every attack move or find every system weakness beforehand.
- Static models: IRS user fixed model that don't always reflect the reality and need subjective guesswork about attack damage and response cost.

Now here the RL helps:

- Adapting to real attack
- Real data

Testbed environment:

In this research work the simulated/virtual environment was created by Graphical Network Simulator 3 (GNS3) software.

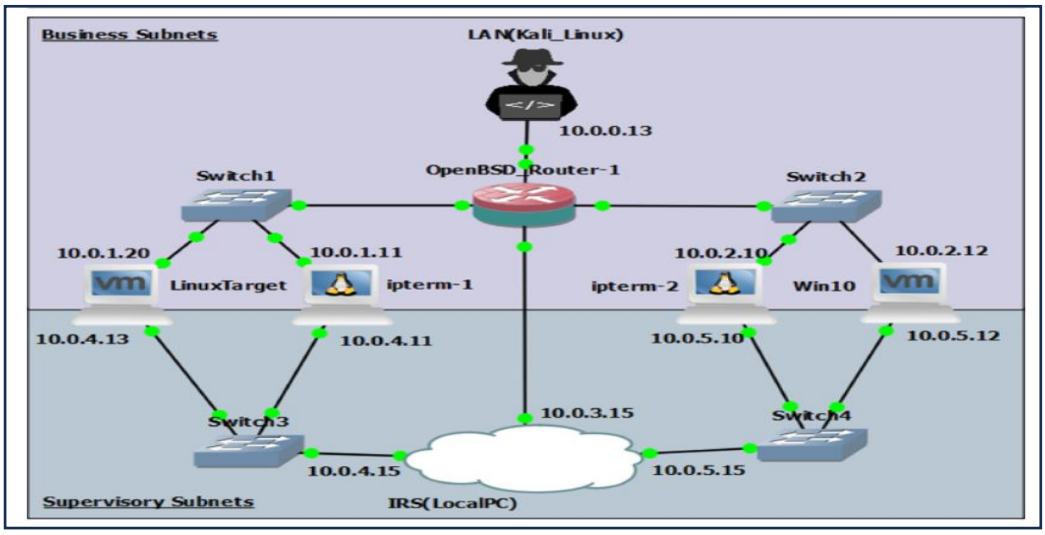
Here two network is created:

- Business network (external LAN): internal network the RL agent is defending.
- Supervisory Network: separated monitoring segment for observing the simulated attack and the RL agent response.

Types of IP address:

- Private: each device connected to LAN is assigned private IP address. It is a local address and can’t be used to go to the internet.
- Class A range from 10.0.0.0 to 10.255.255.255

- Public: It is assigned by our Internet Service Provider (ISP) to our router .



Network Setup:

- OpenBSD, Router and Firewall:
  An OpenBSD virtual machine is configured as the router and firewall, with the pf firewall initially blocking traffic from the external network (WAN) to specific subnet (10.0.2.0/24)
- external LAN (Kali Linux):
  A Kali Linux machine represents the attacker on the external network.
- Internal Network:
  subnet 10.0.1.0/29 - include a metasploitable VM and an IpTerm. node
  subnet 10.0.2.0/24: Include another IpTerm node and W-10 VM.
- Virtualization Environment:
  - VMWare 15 Pro is used to ~~rang~~ manage the VMs.
  - Scripts are used to automate starting, stopping and restoring the VMs to snapshots b/w training episodes.

Challenges:

- Restoring Snapshots: VirtualBox was initially used but encountered issues restoring snapshots, leading to a switch to VMWare.
- Training Speed: Restoring snapshots is time-consuming and limits the training speed of the RL agent.

Tools used:

- Paramiko: A python library handles communication with the VM using some predefined function.
- Tensorforce: A deep RL library used to develop the RL agent (including the PPO agent).

Benefits of this IRS work:

- Able to interact with machine over SSH and collect as much info as the machine's OS can provide.
- This is particularly beneficial for accurate and real-time state parameter collection.

Limitations of this IRS work:

- Memory and requirement to run full VM simultaneously.
- Network traffic simulation is a complex process

In order to effectively integrate a RL solution to an IRS, there is a three challenges are occur in previous related work but these challenges are solve in this research work:

- Translating the network: converting the complex network environment into a continuous state space that the RL agent can understand.
- Defining actions: Choosing a set of descrete actions the RL agent can take to respond to attacks.
- Reward Performance: Designing system to measure the success of the RL agent's actions and translate that into reward for learning

Translating the network to an RL environment:

- In this system we use five core functions for translate network data into an RL-Compatible format. These function deals with observing network state, executing actions, and determining if and episodes should end.
- The functions name is reset(), execute(), getstate(), isTerminal(), getReward().
- Here the episodes management is such that the episodes have a max of 10 timesteps, helping to control training.
- Timeout: A custom function terminates an episodes after 2 minutes 30 seconds to prevent hangs and ensure training speed.
- Here the time limit is set for gain the training efficiency

Attack Scenarios:

To train a RL agent in IRS to defend a network.

Goal:

The attacker aims to steal sensitive data from a windows-10 machine on the network.

Challenges:

The attacker can't directly access the window machine due to firewall.

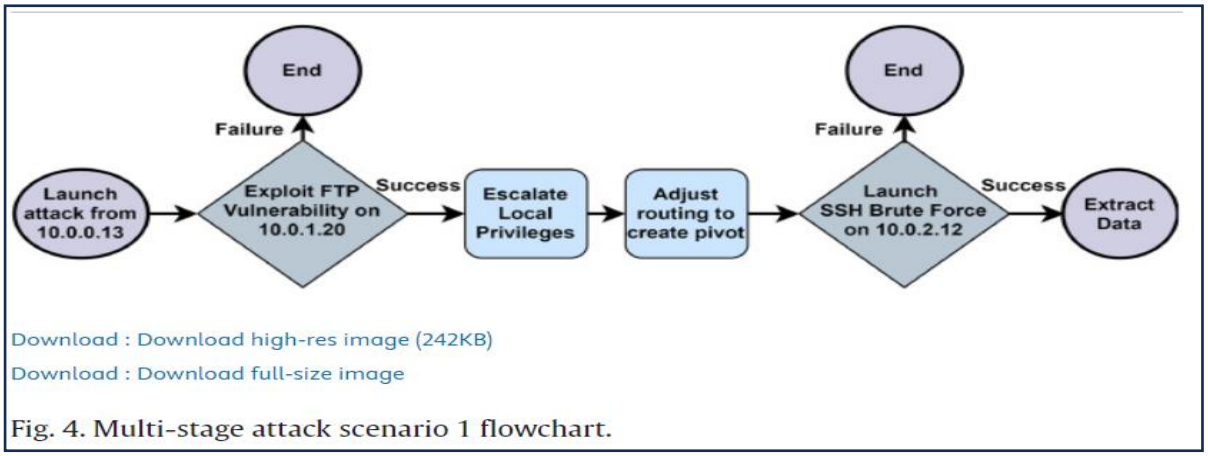
The IRS needs to decide the optimal time and location to respond to the network.

Scenario-1:

- The attacker exploit a vulnerability on an external server (10.0.1.20) to gain access.
- The attacker uses the compromised server as a pivot to reach the window machine.
- The attacker attempts to brute-force SSH on windows machine and extract data.

Scenario-2:

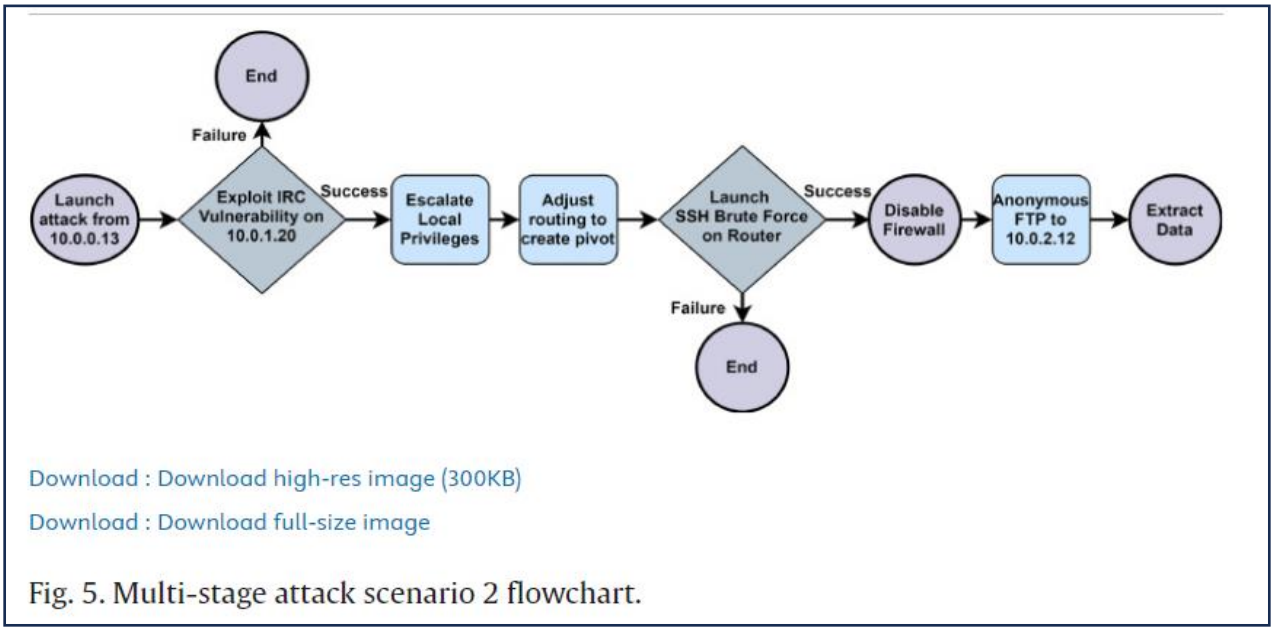
- The attacker exploits a vulnerability on defferent external server.
- The attacker gains access to the router and firewall.
- The attacker disables the firewall and steals data from the windows machine using anonymous FTP.



Download : Download high-res image (242KB)

Download : Download full-size image

Fig. 4. Multi-stage attack scenario 1 flowchart.



Download : Download high-res image (300KB)

Download : Download full-size image

Fig. 5. Multi-stage attack scenario 2 flowchart.

Reward function in model-free RL based IRS:

- Here the RL agent is trained to compute an extrinsic reward based on its observation of a network environment.
- The RL agent received a reward based on the current state of the network (observed using response action functions) and the number of timesteps taken in an episode (one timestep in one action-reward cycle).
- Weights are assigned to different goals (for minimizing alerts) to define their importance in the reward function (algo-1).
- The agent receives a high reward for achieving the main goal.
- The agent can take up to 5 actions per episodes allowing it to explore different response strategies.
- The algo-2 calculate the rewards based on the state and the number of timesteps.

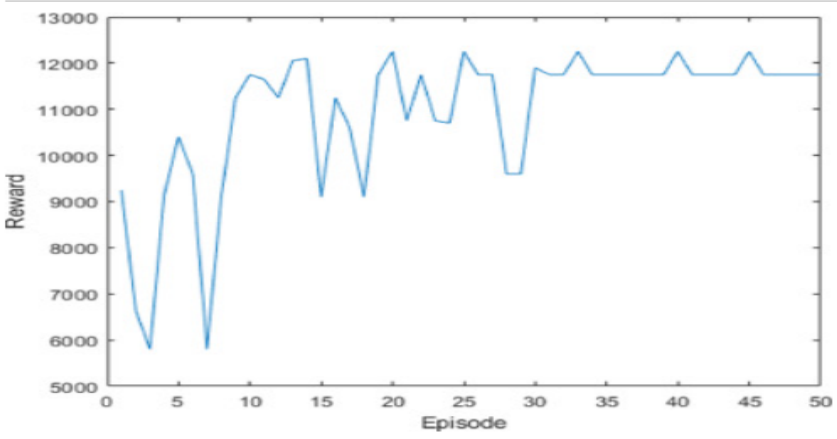
Algorithm 1 (positive reward):

The agent receives a high reward for achieving the goal (stopping the attack) and maintaining a healthy network state.
Smaller rewards are given for intermediate steps.

Algorithm 2 (negative reward):

The agent starts with a high reward, which is then decreased based on negative events (e.g., new alerts, down nodes).

The below figure shows that the agent using Algorithm 2 performs better and converges faster towards an optimal policy.



Download : Download high-res image (131KB)
Download : Download full-size image

Fig. 8. Update frequency 1, negative reward function.

Algorithm 1. Reward evaluation

```
Require: state, steps
reward=state.nodesUp * 50
reward=reward – (state.packetErrors)
If state.alerts < 1 Then
reward=reward * 1.5
Else
reward=reward * 0.4
If state.nodesUp=5 And state.alerts=0 Then
reward=reward * 4
reward=(1 - (steps / 10)) * reward
Return reward
```

Algorithm 2. Negative reward evaluation

```
Require: state, steps
reward=3000
If state.alerts >= 1 Then
reward=reward – (state.alerts * 500)
If state.nodesDown >= 1 Then
reward=reward – (state.nodesDown * 500)
If state.packetErrors >= 1 Then
reward=reward – (state.packetErrors * 50)
reward=reward – (steps * 150)
If reward < 0 Then
reward=0
Return reward
```

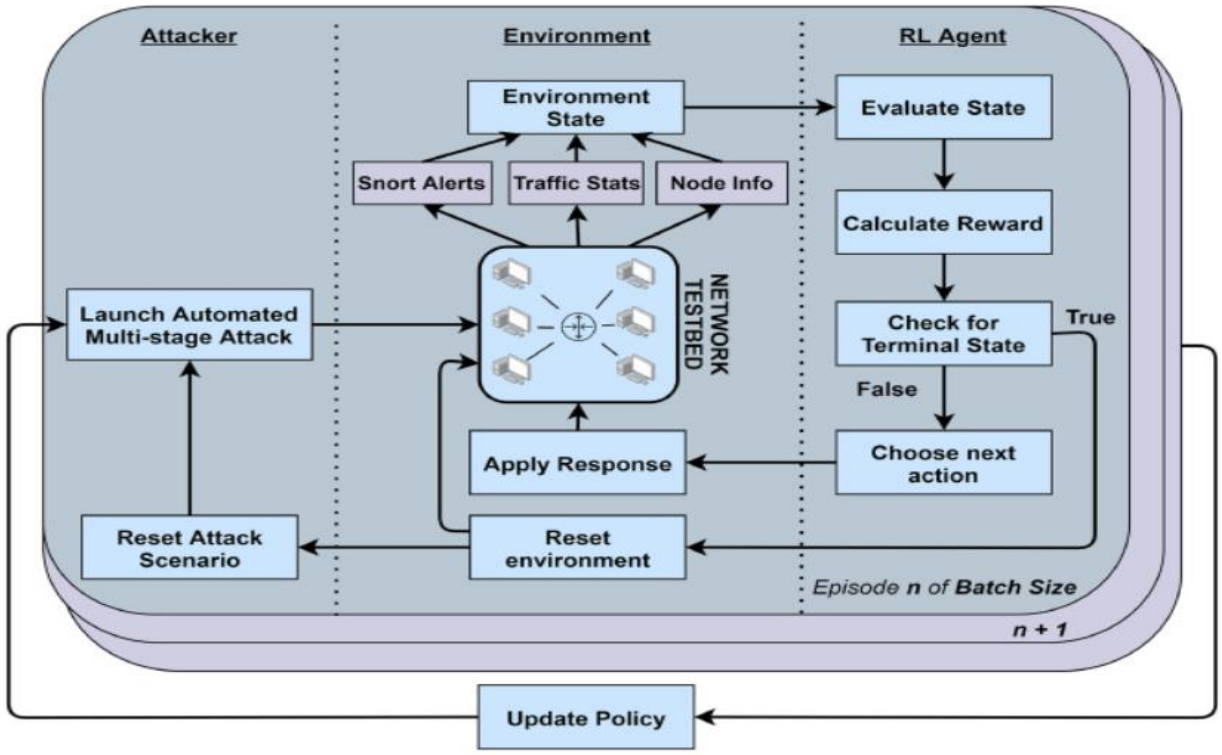
Feature	Algorithm 1 (Positive Reward)	Algorithm 2 (Negative Reward)
Initial reward	High	High
Reward for good actions	Increases reward	No change in reward
Reward for bad actions	No change in reward	Decreases reward

1. Positive reward:

The agent receives a reward for taking actions that lead to desirable outcomes.

2. Negative reward:

The agent starts with a high reward and is penalized for taking actions that lead to undesirable outcomes.



Download : Download high-res image (605KB)

Download : Download full-size image

Fig. 6. RL training framework.

Training Framework:

For train our agent in the given loop we focus only on three things:

1. Episodes:

The system runs multiple training episodes (specified by the batch size) before update the RL agent's policy.

2. Time step:
   Within each episode the environment goes through multiple timestep. During each timestep the state of the network is observed through 3 variables

- Node info: Node status (up/down)
- Traffic Statistics:
- Snort alerts: Security alerts from the Snort IDS.

The RL agent select an action based on the observed state.
The chosen action is applied to the network environment.
The reward is calculated based on the new state (e.g., higher reward for stopping the attack).

3. Policy Update:
   After enough episodes the RL agent's policy (the strategy for choosing action) is updated based on the collected experience (rewards and states).

Experimentation with model-free RL based IRS:

Slow Simulation:

Using VM to simulate the network environment makes resulting the environment time-consuming, leading to longer training times compared to simple RL experiments.

Adapting training:

- Limited episodes: here the training has to be done with fewer episodes due to the lengthy reset times.
- Hyperparameter Tuning: tune the learning rate.

Reward Dynamics:

- Initial high rewards: the custom reward function (algo-2) starts the agent with higher reward.
- Significant drops in reward likely shows that the multiple stage attack succeeding.
- Default VS Adjusted: The default PPO learning rate (1e-3) makes learning too slow for this scenario, causing poor convergence within the limited episodes.

Choosing the right approach depends on several factors, including:

The specific goal of the RL agent:

In the case of an Intrusion Response System (IRS), the goal is to stop attacks. Penalizing the agent for negative actions (like failing to stop the attack) is simpler and more effective than rewarding successful actions.

The complexity of the environment:

Defining a positive reward function can be challenging, especially for complex environments with many possible states.

The desired learning behavior:

Negative rewards can encourage the RL agent to avoid specific actions more effectively.

Hyperparameters	Description	Experimented Range
Episode number	Defines the number of episodes of training to complete. More episodes result in longer training.	30 - 150
Batch size	The number of episodes used in a policy update to adjust the underlying neural net.	2 - 8
Update frequency	Defines the number of episodes after which a policy update occurs.	1 - 4
Learning rate	Defines how the extent to which new experience overrides previous experience.	1e-3 – 1e-1

Solution:

Hyperparameters were adjusted to compensate for the limited training data:

- Batch size: Reduced from 32 to a smaller value. This is speeds up training but can make results less stable.
- Update frequency: Increased from the typical value of 1 to 4. This allows the agent to learn from its experiences more frequently.

The chosen values (batch size: 6, update frequency: 3, learning rate: 1e-2) were found to be optimal for this specific scenario.

Benefits, Challenges, and future work for model-free RL based IRS:

Benefits:

- Removes the need for subjective values like "response cost" by learning from direct experience.
- Adapts to changing situations and considers attack severity when evaluating responses.
- Handles complex multi-stage attacks using real attack scenarios.
- Uses a larger set of response actions compared to limited or attack-specific sets.

Challenges:

- Limited training episodes due to slow virtual machine simulations.
- Difficulty in comparing performance with other IRS proposals due to the lack of standardized testbeds.
- Requires a large number of training episodes to fully achieve the potential of generalizing to new attack scenarios or network changes.

Future work directions:

Scalability:

- Utilize faster technologies like containers or parallelization to reduce simulation time and enable more training episodes.
- Explore standardized testbeds for better comparison with other IRS proposals.

Generalizability:

- Investigate the ability of the RL agent to adapt to new attacks or network changes.

Distributed IRS:

- Train multiple RL agents, each specializing in handling specific types of attacks, like tactics or techniques from the MITRE ATT&CK framework.

-

-