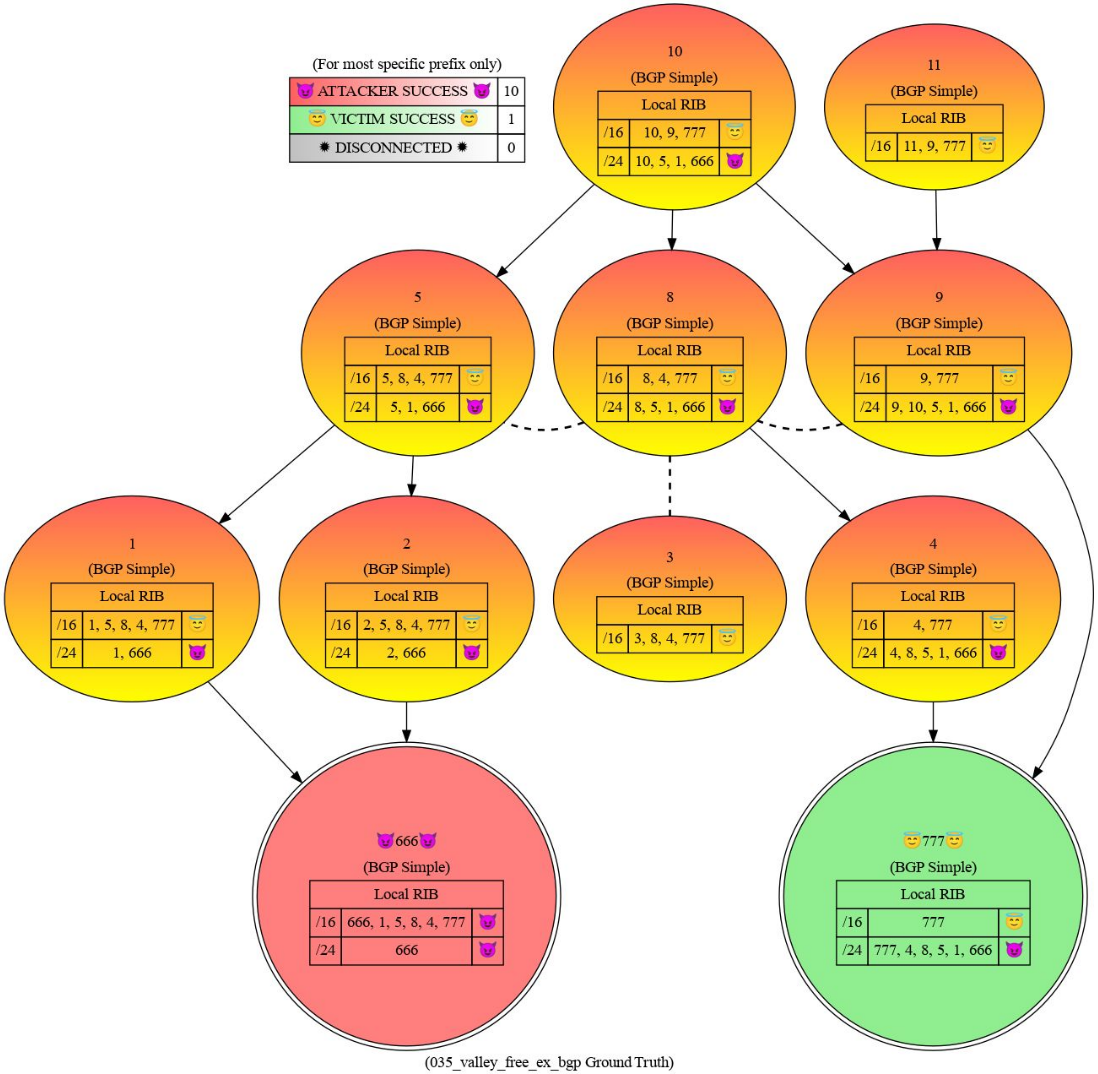
BGPy:
The BGP Python Security Simulator

By: Justin Furuness, Cameron Morris, Reynaldo Morillo, Dr. Amir Herzberg, Dr. Bing Wang

(Affiliated with the University of Connecticut)

Background

- The internet is from a high level, a directed acyclic graph of over 70k nodes, known as Autonomous Systems, or ASes for short
  - These ASes route traffic for ISPs, CDNs, Organizations, etc
  - They can be identified by their AS Number, or ASN for short

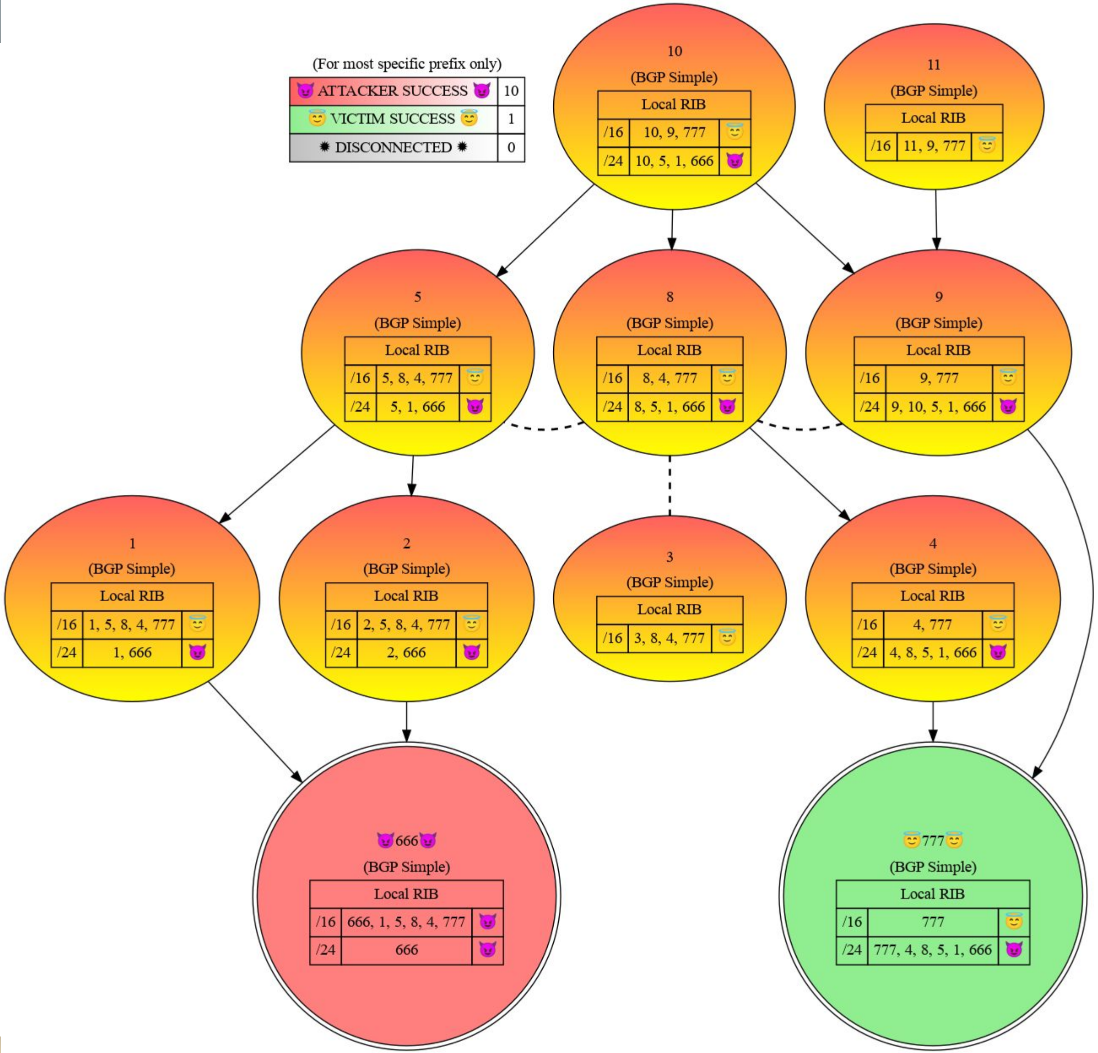


 github.com/jfuruness/bgpy

AS Relationships

- For our simulations, we focus on two of the most prevalent type of connections between ASes:
  - Provider -> Customer connections
    - Customers pay provider
  - Peer <-> Peer connections
    - Traffic flows freely
  - Sibling relationships not included

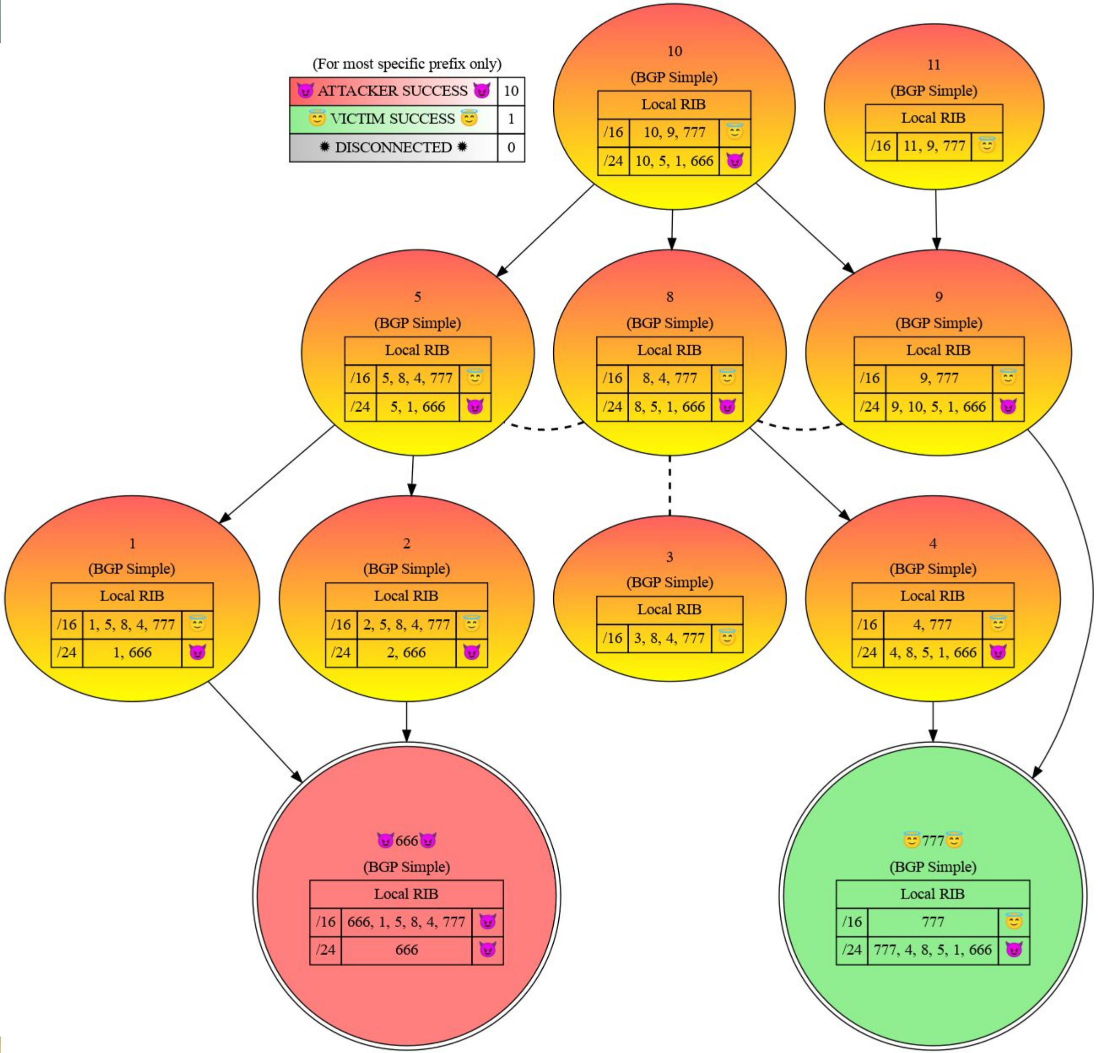
03 github.com/jfuruness/bgpy



(035_valley_free_ex_bgp Ground Truth)

Announcements

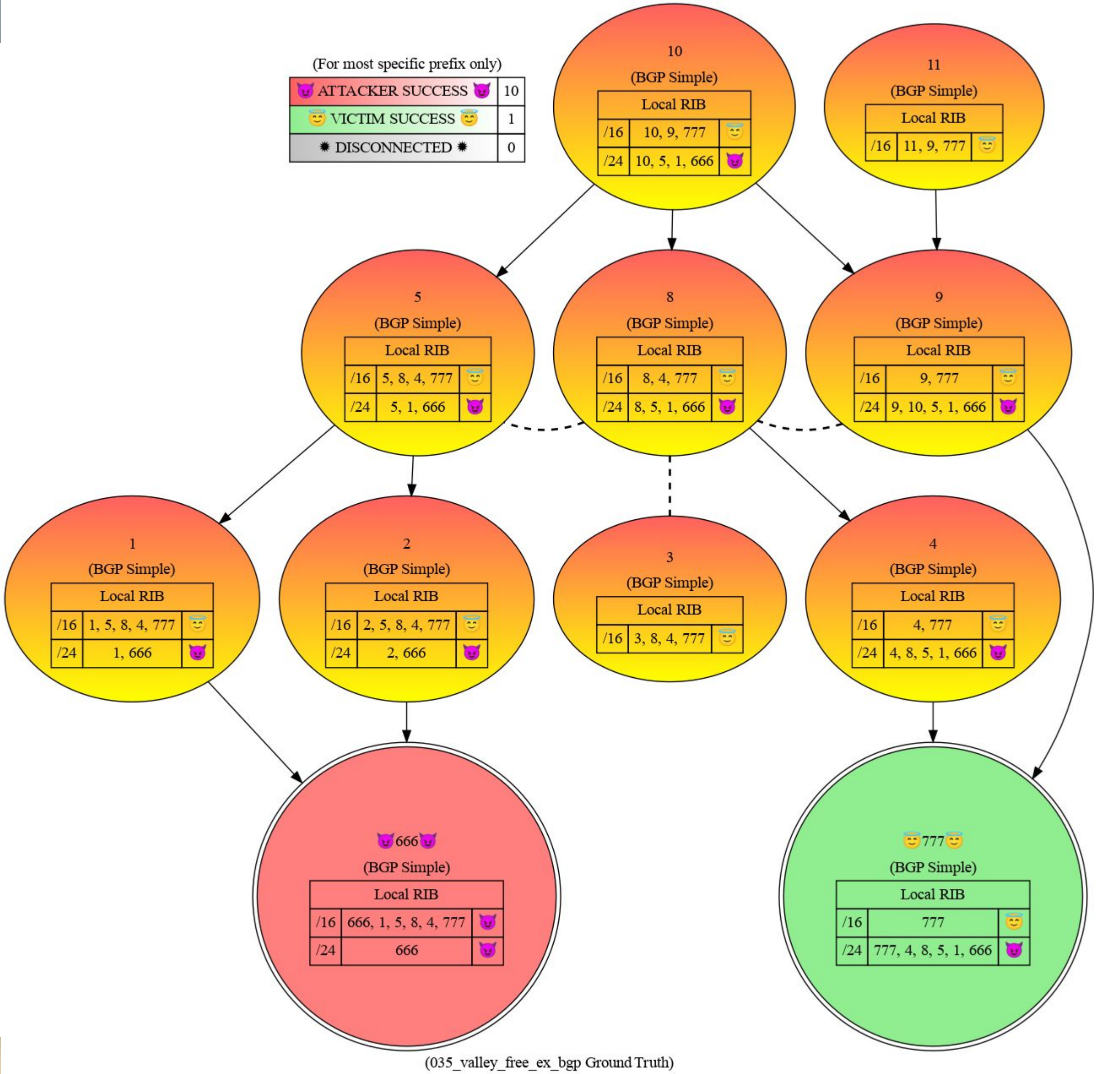
- ASes forward traffic from one another. While announcements have many attributes, from a high level we can think of them as:
    - Prefix (containing a block of IP addresses), such as 1.2.3/24
    - Origin (the ASN that created the announcement)
    - AS Path (The path that the announcement took to reach it's destination



(035_valley_free_ex_bgp Ground Truth)

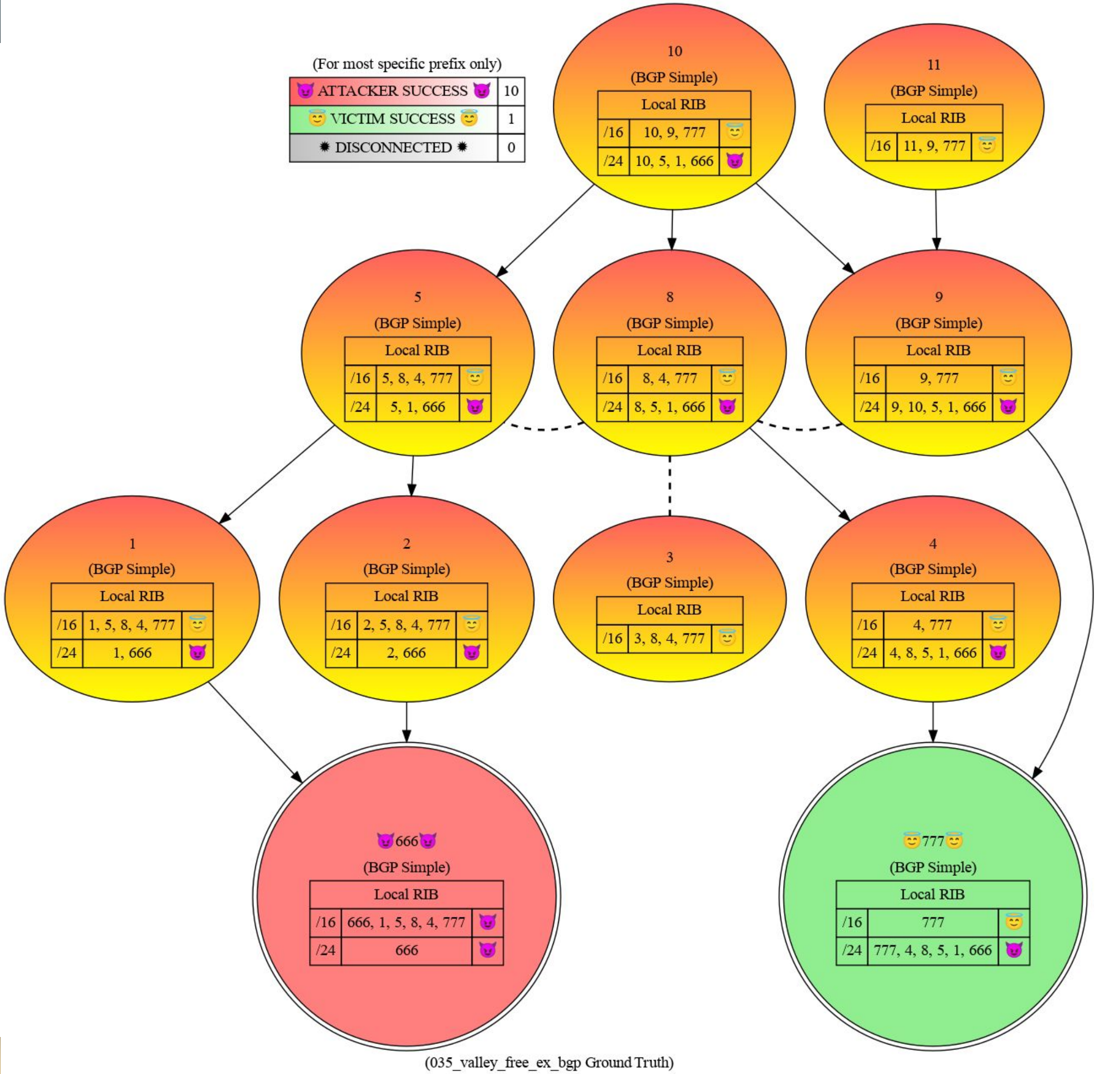
Valley Free Routing

- Valley Free Routing is the default in our simulations, similar to prior works. This selects the best announcement in order to maximize profit.
  - customers > peers > providers
  - Shortest AS Path
  - Tiebreakers
    - Default to lowest ASN



Export Policy

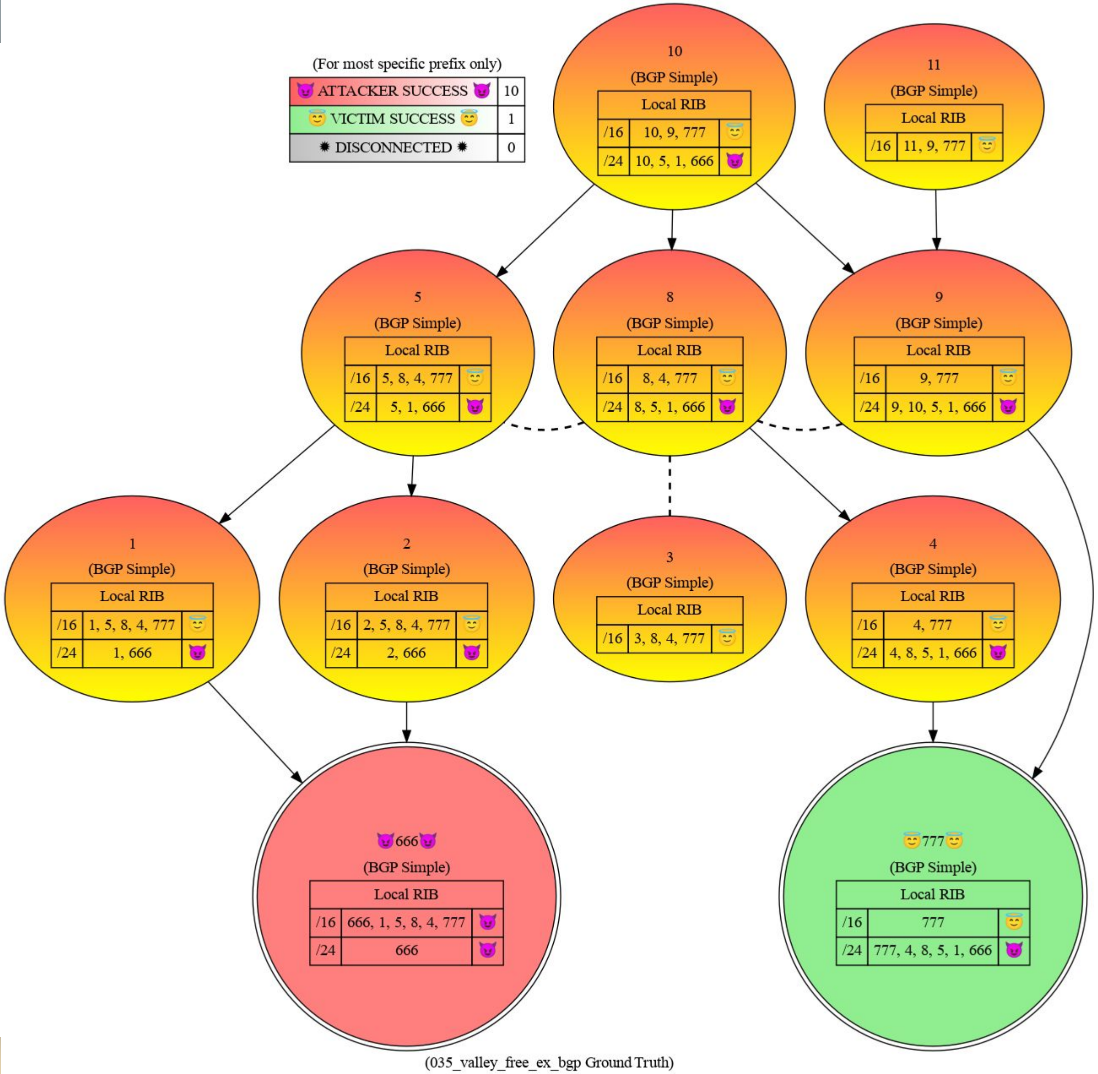
- Export Policy:
  - Announcements received from customers are sent to all
  - Announcements received from peers and providers are sent to customers



 github.com/jfuruness/bgpy

BGP Hijacking

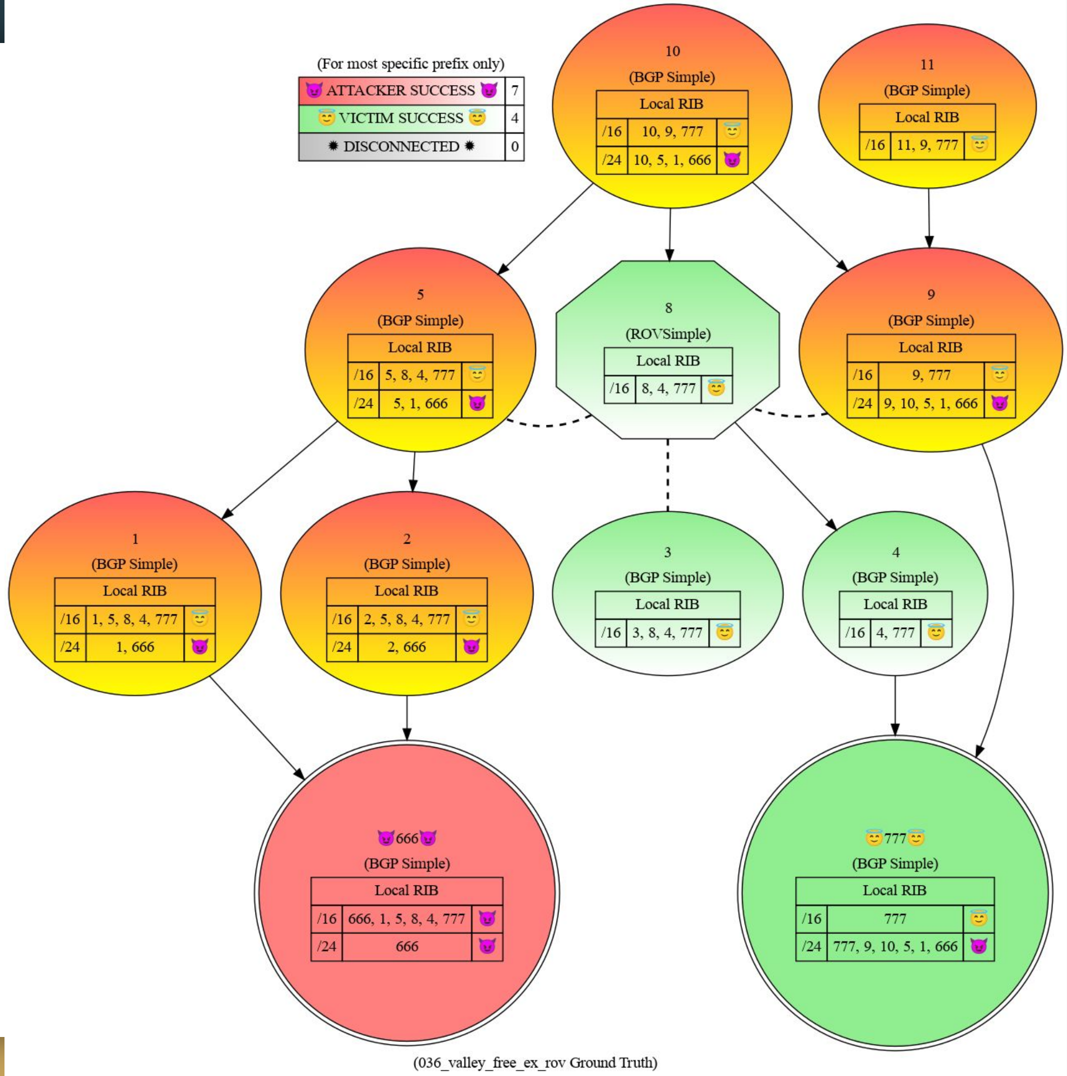
- How does data plane traffic flow work if you have multiple prefixes for the same IP address?
  - Most specific prefix is chosen
  - This allows for BGP hijacks
  - This specific example is of a subprefix hijack



07 github.com/jfuruness/bgpy

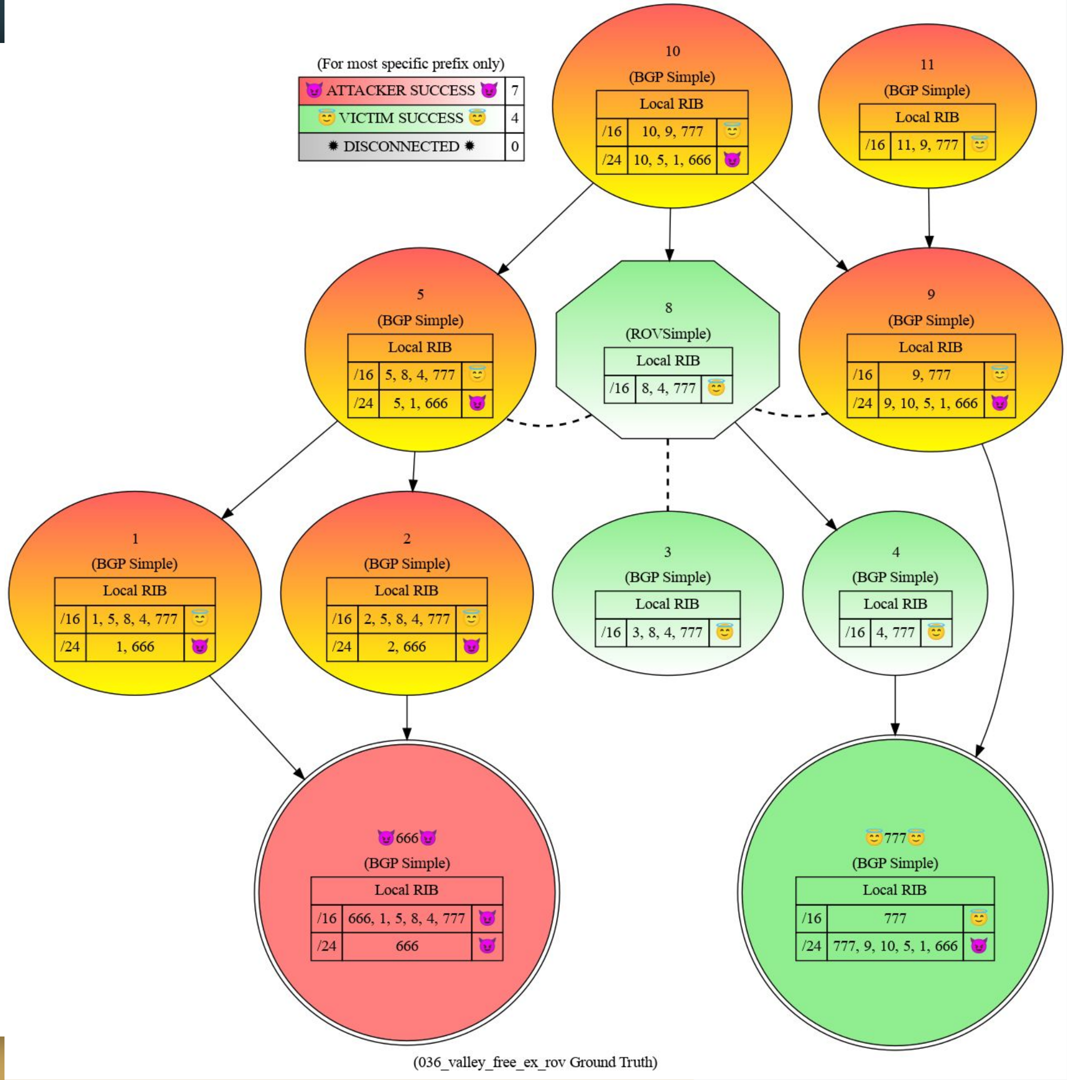
ROV

- From a very high level, Route Origin Validation (ROV) can declare announcements as valid or invalid
- Here AS 8 deploys ROV
- Since ROV declares the /24 subprefix invalid, AS 8 drops the /24 prefix. This protects:
  - AS 3
  - AS 8
  - AS 4



BGPy

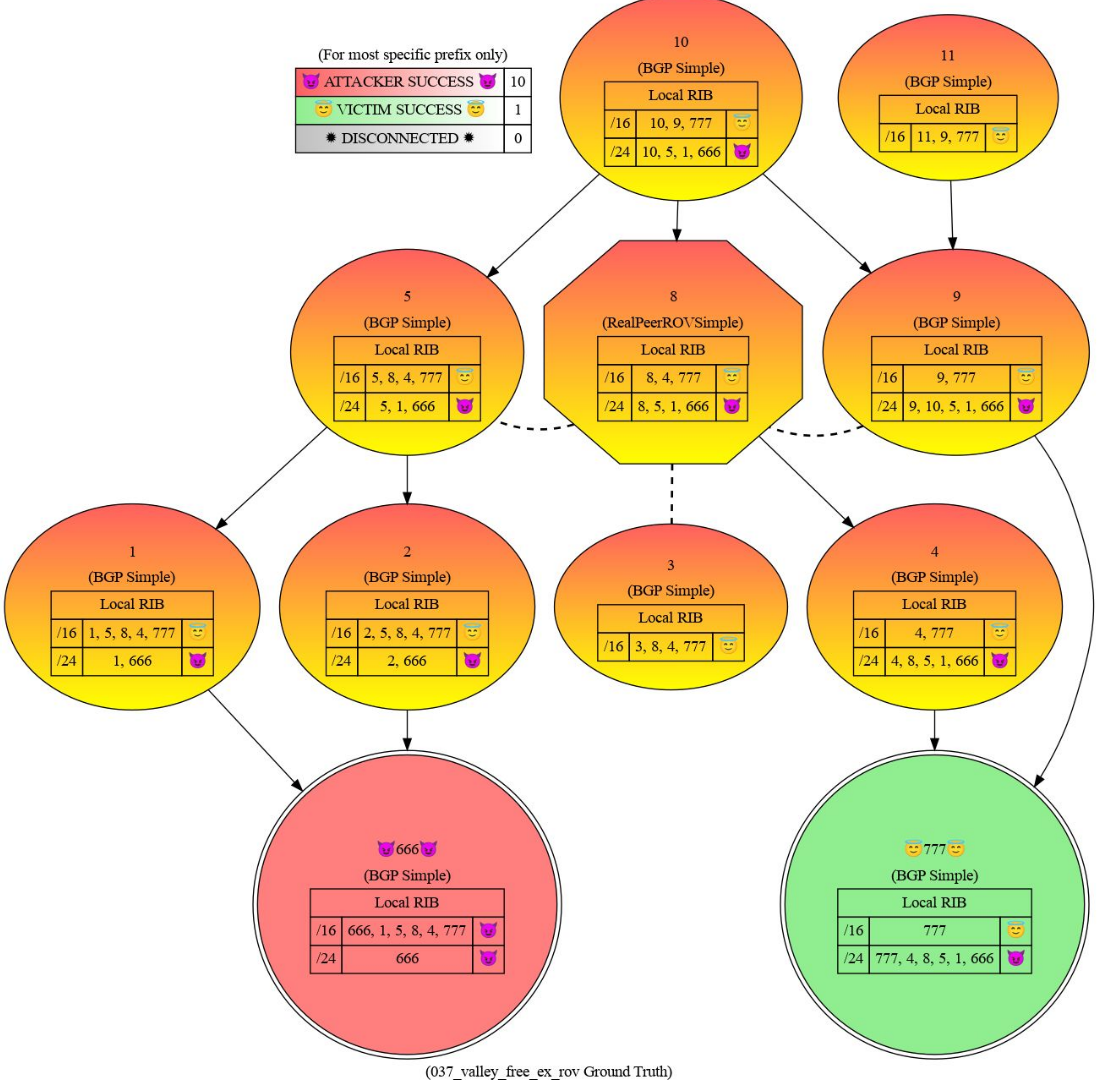
- This is a system test included in BGPy
  - Easy to write
  - Easy to view
  - Easy to verify
- On a much larger scale, BGPy is open source and can simulate different attacks and defenses that are partially adopted
- The results can be compared with well defined benchmarks
- It's efficient enough to be run on a laptop and is easy to extend



 github.com/jfuruness/bgpy

PeerROV

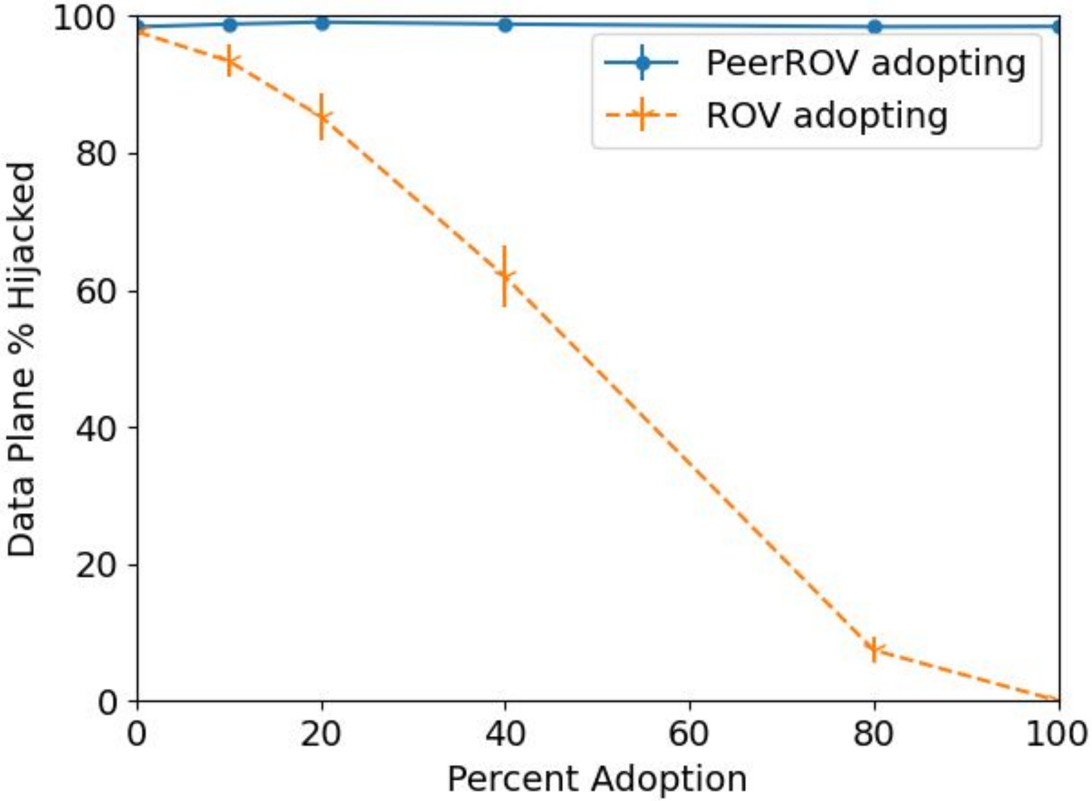
- PeerROV is an ROV variant that only filters peers
- Here AS 8 deploys Peer ROV
- Peer ROV does not protect any ASes in this case



 github.com/jfuruness/bgpy

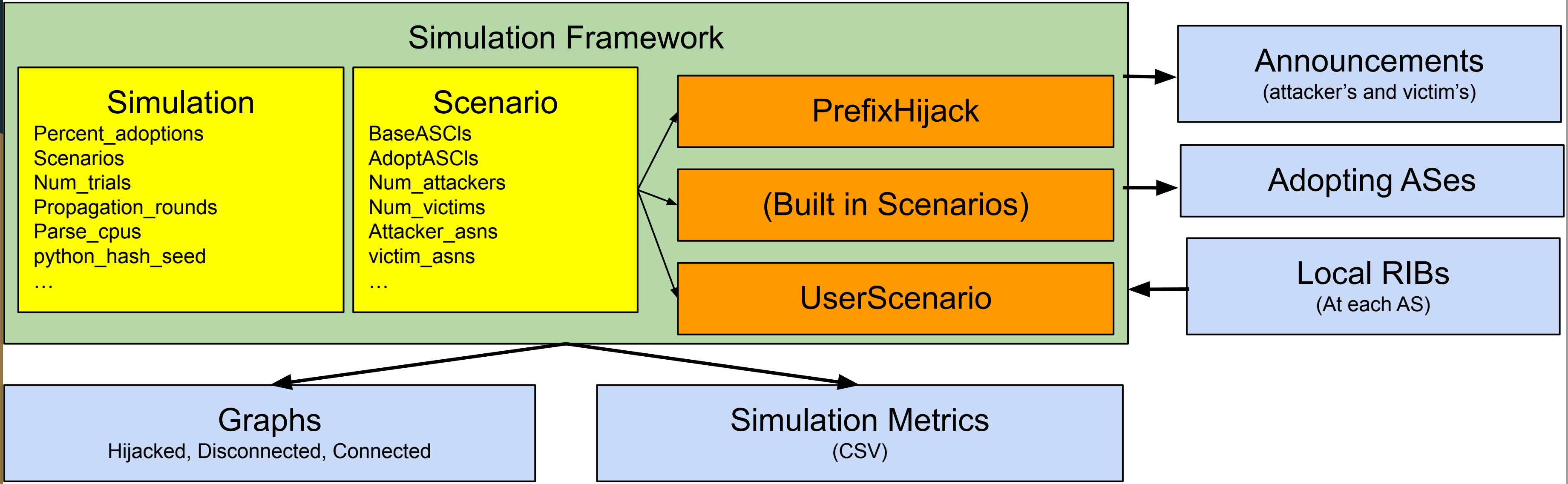
Results

- This is an example of a graph included by default in BGPy
- Here we are comparing ASes th adopting ROV vs PeerROV
- For example, at 40% adoption:
  - About 60% of ROV ASes are hijacked
  - Almost all PeerROV ASes are hijacked
- We also track
  - Percent Disconnected
  - Percent Successfully Connected



BGPy: The Simulation Framework

The Simulation Framework is a wrapper around the Simulation Engine that facilities the comparison of multiple security policies against attack scenarios at partial adoptions



 github.com/jfuruness/bgpy

BGPy: The Simulation Framework (PseudoCode)

```
engine = SimulationEngine()
metric_tracker = MetricTracker()
for trial in range(num_trials):
    for scenario in scenarios:
        # Select attacker(s) and victim(s), kept consistent across scenarios
        scenario.select_attacker_and_victim(engine)
        # Select Adopting ASes, kept consistent across scenarios
        # Also sets Adopting ASes, which varies from scenario to scenario
        scenario.set_adopting_ases(engine)
        # Seeds the attacker(s) and victim(s) announcement in the engine
        scenario.seed_attacker_victim_announcements(engine)
        # Propagates announcements throughout the AS topology
        engine.run()
        # Records metrics for graphing later
        metric_tracker.analyze(engine)
        # Remove announcements from the graph
        engine.clear()
```

 github.com/jfuruness/bgpy

```python
class SubprefixHijack(Scenario):
    def _get_announcements(self, *args, **kwargs):
        anns = list()
        anns.append(
            self.scenario_config.AnnCls(
                prefix=Prefixes.PREFIX.value,
                as_path=(victim_asn,),
                roa_valid_length=True,
                roa_origin=victim_asn,
            )
        )

        anns.append(
            self.scenario_config.AnnCls(
                prefix=Prefixes.SUBPREFIX.value,
                as_path=(attacker_asn,),
                roa_valid_length=False,
                roa_origin=victim_asn,
            )
        )
        return tuple(anns)
```

BGPy: Scenario

The Scenario controls attacker and defender strategies, including:

- Which ASes can be chosen as attacker/victim
- What announcements those ASes announce
- What routing policies are deployed across the adopting ASes
- Etc

 github.com/jfuruness/bgpy

BGPy: Simulation

The Simulation object controls all aspects of the simulation

- Trials, CPUs, etc
- Percent Adoptions
- Scenarios

```
sim = Simulation(
    percent_adoptions=(
        SpecialPercentAdoptions.ONLY_ONE,
        0.1,
        0.2,
        0.4,
        0.8,
        SpecialPercentAdoptions.ALL_BUT_ONE,
    ),
    scenario_configs=(
        ScenarioConfig(ScenarioCls=SubprefixHijack, AdoptASCls=ROVSimpleAS),
        ScenarioConfig(ScenarioCls=SubprefixHijack, AdoptASCls=PeerROVSimpleAS),
    ),
    output_dir=Path("~/Desktop/paper_graphs").expanduser(),
    num_trials=1000,
    parse_cpus=12,
)
sim.run()
```

 github.com/jfuruness/bgpy

BGPy: ROV AS

ROV AS is an example of how you would subclass the default BGP AS

- Most extensions are fairly simple
- Here we merely extend the valid announcement check (which checks for loops in the AS path)

```python
class ROVAS(BGPAS):
  def _valid_ann(self, ann: Ann) -> bool:
    # If ROA is invalid, ROV says announcement is invalid
    if ann.invalid_by_roa:
      return False
    # If ROA is valid, determine validity with BGP
    else:
      return super(ROVAS, self)._valid_ann(ann)
```

 github.com/jfuruness/bgpy

BGPy: Peer ROV AS

Peer ROV is the same as ROV, except it only filters announcements from peers

Again fairly straightforward in BGPy

```
class PeerROVAS(BGPAS):
  def _valid_ann(self, ann: Ann) -> bool:
    # If ROA is invalid, ROV says announcement is invalid
    # For PeerROV, only filter by Peers
    if ann.invalid_by_roa and ann.recv_relationship == Relationships.PEERS:
      return False
    # If ROA is valid, determine validity with BGP
    else:
      return super(PeerROVAS, self)._valid_ann(ann)
```

 github.com/jfuruness/bgpy

BGPy: Announcement

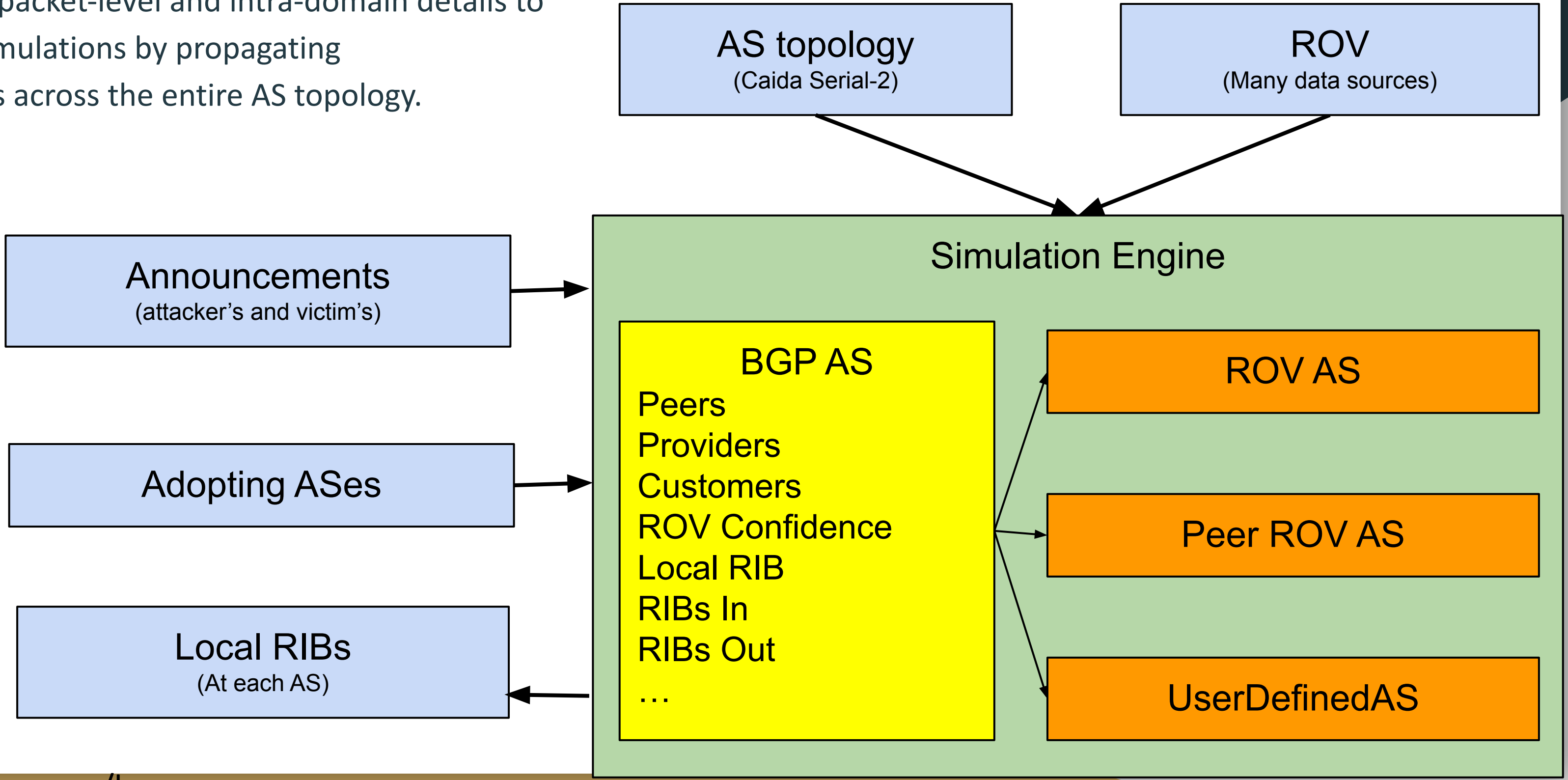
Sometimes it’s desired to add extra attributes to announcements

Here is an example for ROV++, where we added a few attributes

```
@dataclass(frozen=True, slots=True)
class ROVPPAnn(Announcement):
    holes: tuple[str] = ()
    blackhole: bool = False
    # V3 attributes
    preventive: bool = False
    attacker_on_route: bool = False
```

BGPy: The Simulation Engine

Abstracts away packet-level and intra-domain details to perform BGP simulations by propagating announcements across the entire AS topology.



 github.com/jfuruness/bgpy

BGPy: The Simulation Engine (ROV Data)

- Optionally, real world ROV data can be used in the simulations
- Data Sources included by default:
  - Rov.rpki.net
  - Isbgpsafeyet.com
  - Revisiting RPKI Route Origin Validation on the Data Plane

BGPy: The Simulation Engine (Propagation)

- First, announcements are inserted at the victim(s) and attacker(s)
- Then, announcements are propagated throughout the AS topology:
  - First from customers to providers
  - Second from peer to peer
  - Third from providers to customers
- We converge in O(E) time in a single round of propagation
  - This allows us to write BGPy in Python and still be able to run it on a laptop
  - If users would like to propagate for more than 1 round, they can set the number of rounds in a parameter

BGPy: Future Work

- We just released a new version with some improvements, and will continue to iterate
- The next big leap forward will be likely using rust bindings with PyO3 for massive speed improvements
- More real world data can be added to make simulations more realistic
    - IXP data
    - BGP Communities
    - AS Blacklists
    - etc

Thank you! Questions?

- Contact Info:
  - Justin Furuness: jfuruness@gmail.com
  - Cameron Morris: cameron.morris@uconn.edu
  - Reynaldo Morillo: reynaldo.morillo@uconn.edu
  - Dr. Amir Herzberg: amir.herzberg@gmail.com
  - Dr. Bing Wang: bing@uconn.edu
- Link to BGPy: github.com/jfuruness/bgpy