Advances in Automated Attack Planning

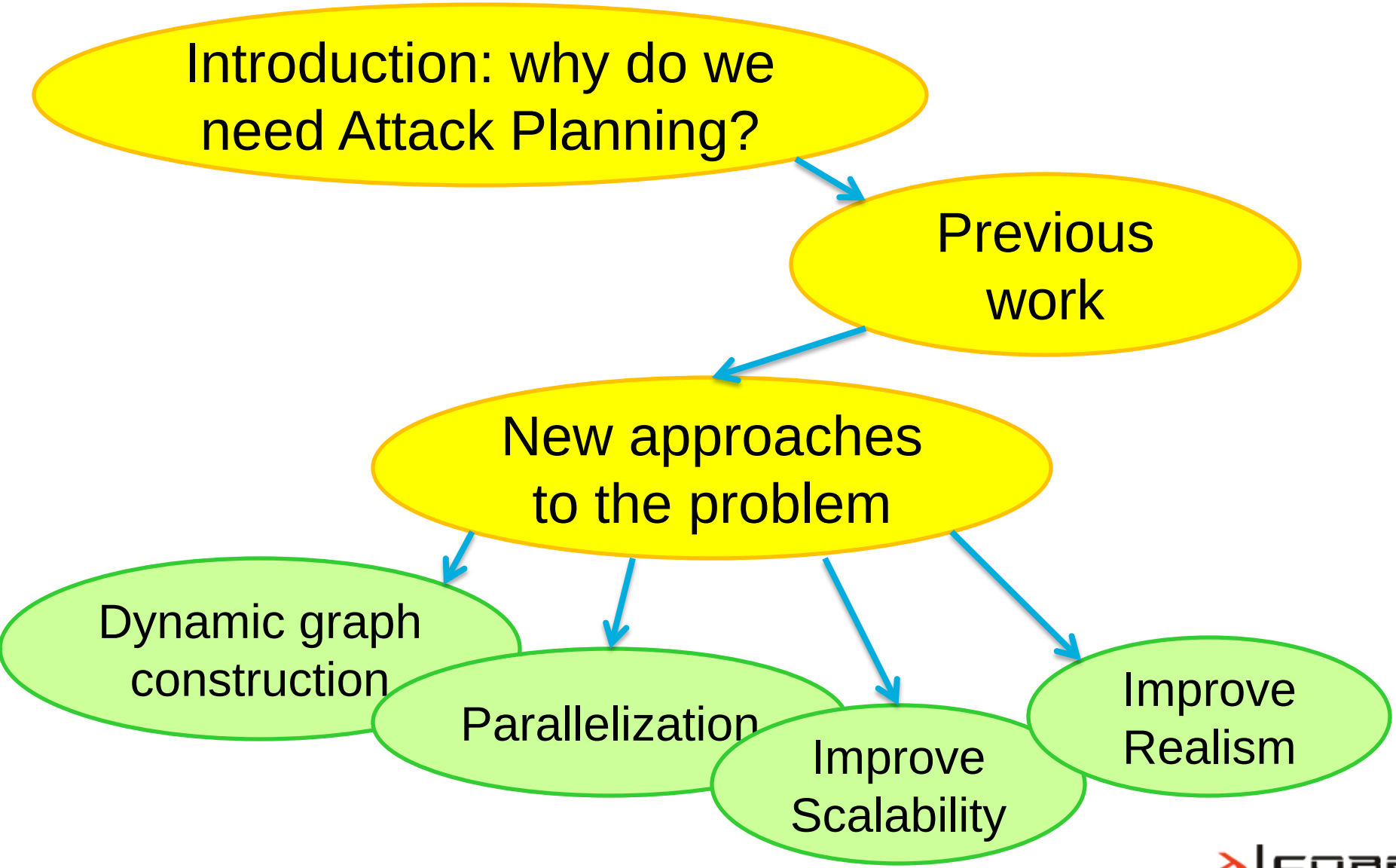
Alejandro David Weil [1]

Carlos Sarraute [1,2]

1. Core Security Technologies 2. Ph.D. in Informatics Engineering, ITBA

PacSec - Nov 12/13, 2008

Agenda


Introduction: why do we need Attack Planning?
Previous work
New approaches to the problem
Dynamic graph construction
Parallelization
Improve Scalability
Improve Realism

Introduction

Why do we need automation?

- Evolution of pentesting
  - Attacks are evolving
  - Organizations are evolving
    - technological complexity
    - infrastructure complexity
  - Manual pentesting requires more expertise and time
  - Continuous pentesting
- Make security testing more accessible
  - So the BOFH can test his own network

Increase pentesting scale

- Example: pentest a network with 200 machines
  - limited human resources
  - bounded time frame
  - pentest mimics attacks which doesn't have those restrictions
- Automate repetitive tasks
- Liberates time for
  - research / creative work
  - training / be up-to-date
  - produce more complex attacks

Brute force is not enough

- Different types of tasks
  - Dependencies between tasks
  - Tasks have to be prioritized
- We need a strategy

→ This is a planning problem!

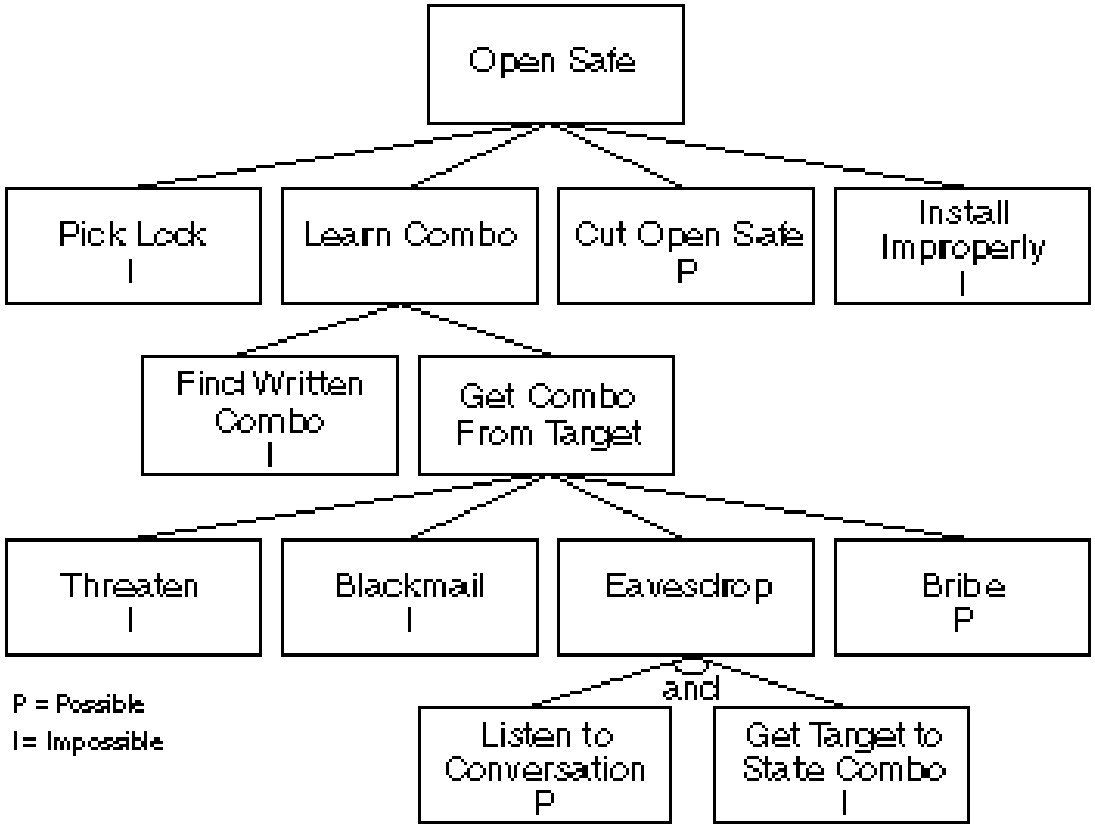
The Planning Problem

- Given a set of goals, and an initial incomplete knowledge of the network, continuously determine the best course of actions for an attacker in order to obtain the goals
  - execute the actions
  - feedback the plan with their results

Other Models

Attack Trees

- Introduced by Bruce Schneier (1999)



Attack Graphs

- Cynthia Phillips and Laura Swiler (1998)
  - A graph-based system for network-vulnerability analysis
- Oleg Sheyner and Jeannette Wing (2003)
  - Tools for generating and analyzing Attack Graphs
- Steven Noel and Sushil Jajodia (2004)
  - Managing Attack Graph complexity through visual hierarchical aggregation
- R. Lippman and K. Ingols (2005)
  - Review of papers on Attack Graphs

Basic model

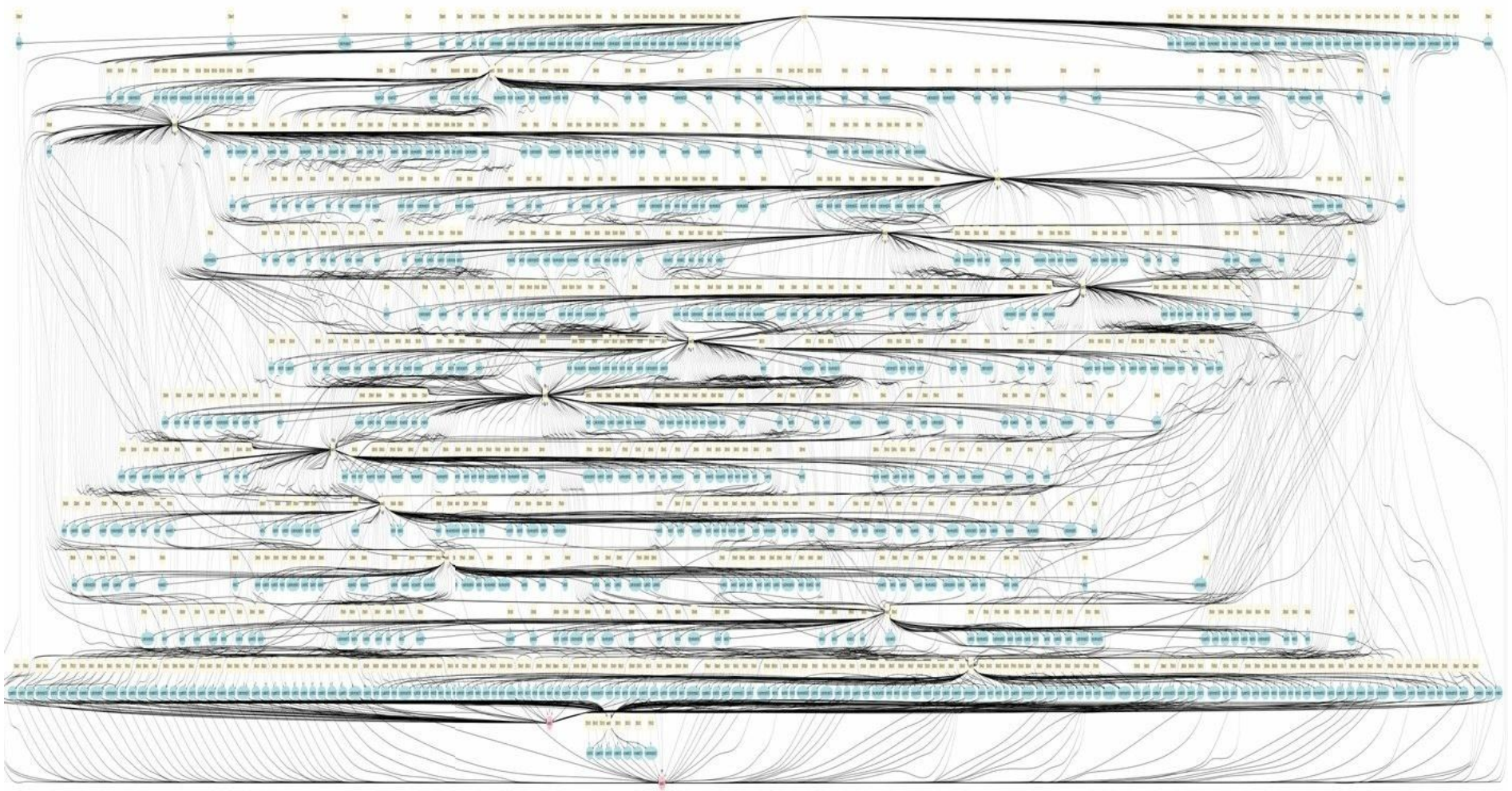
- The graph nodes are actions and predicates.
- An action has:
  - preconditions / requirements
  - effects / results
- Preconditions are predicates that must be true before the action is executed.
- Effects are predicates that become true after the action is executed.

Example of an action

- Action IIS-buffer-overflow
  - preconditions
    - privileges(S) ≥ user
    - privileges(T) < root
    - IIS_service(T)
    - reachable(S, T, 80)
  - effects
    - privileges(T) := root
    - not IIS_service(T)

From Sheyner – Wing "Tools for generating and analyzing Attack Graphs"

Example of Attack Graph



- From Noel – Jajodia: "Managing Attack Graph Complexity Through Visual Hierarchical Aggregation"

Limitations (1)

- Constructed from a network defender's point of view
  - suppose a complete and intimate knowledge of the network from the beginning
- Build a static plan
- Only consider exploits…
- …without executing them!
  - find a sequence of potential exploits to reach a goal, not real vulnerabilities
  - generate false positives

Limitations (2)

- Small size of solvable scenarios
  - many papers build the complete attack graph
  - combinatorial explosion as the depth of the graph increases
  - in general, solvable scenarios have 10 machines and 10 exploits or less
- Additional hypothesis are introduced to reduce complexity
  - hierarchical aggregation
  - monotonicity of the attack

RPT
A practical and sound approach

RPT in real scenarios

- RPT stands for "Rapid Penetration Testing"
  - implemented in Impact, has 6 years of evolution

Modules by category

Category	Modules
Remote Exploits	177
Local Exploits	61
Client-side Exploits	140
Denial-of-Service (DoS) Exploits	27
Utilities	158
Total	**563**

Target entry points

Operating System	Exploits	Unique Targets
Windows Vista	42	116
Windows 2003	113	743
Windows XP	216	1236
Windows 2000	229	2403
Windows NT	19	84
Linux	155	478
Solaris	32	90
AIX	3	5
Mac OS X	9	53
OpenBSD	15	41
FreeBSD	7	17
Total	**840**	**5266**

- Deals with 840 exploits, targeting 5266 unique targets
- Tested on Class B networks with 512 hosts

The RPT graph

- We build a graph with nodes representing Goals of different kinds connected by their dependencies
  - Dependencies can be generated in runtime
  - Connected graph components are subplans that run in parallel
- The building blocks are Goals, Strategies, Actions and the Planner

Goals

- There are different kind of goals representing any objective / information we are interested in.
- Goals start in *Unknown* state and can be resolved to *Valid* or *Invalid*.
  - goals are resolved by the execution of an action
- A Goal requirements are other Goals
  - can be serial or parallel

Some basic Goals

- Examples:
  - OS of a Host
  - Services of a Host
  - Agent on a Host
  - Agent on a Network
  - IG (information gathering) on a Network
  - DoS of a Host
  - DHCP Service disabled

Goal Quantifiers

- Reduce the complexity of the graph by grouping atomic goals
- The quantifiers are: Any, All, AllPossible
- Examples:
  - Compromise **Any** Host in the net: 53.55.66.0/23 (at least one)
  - **All** TCPPorts from: (21,22,23,80)
  - **AllPossible** Hosts from: 192.168.1.0/24

Strategy

- To resolve a goal we need to know which actions will fulfill it
  - it can be an exploit, a network discovery module, an OS detection module, etc.
- We call *Strategy* the component in charge of mapping goals to actions.
- Designed to easily modify or write new strategies with different purposes.

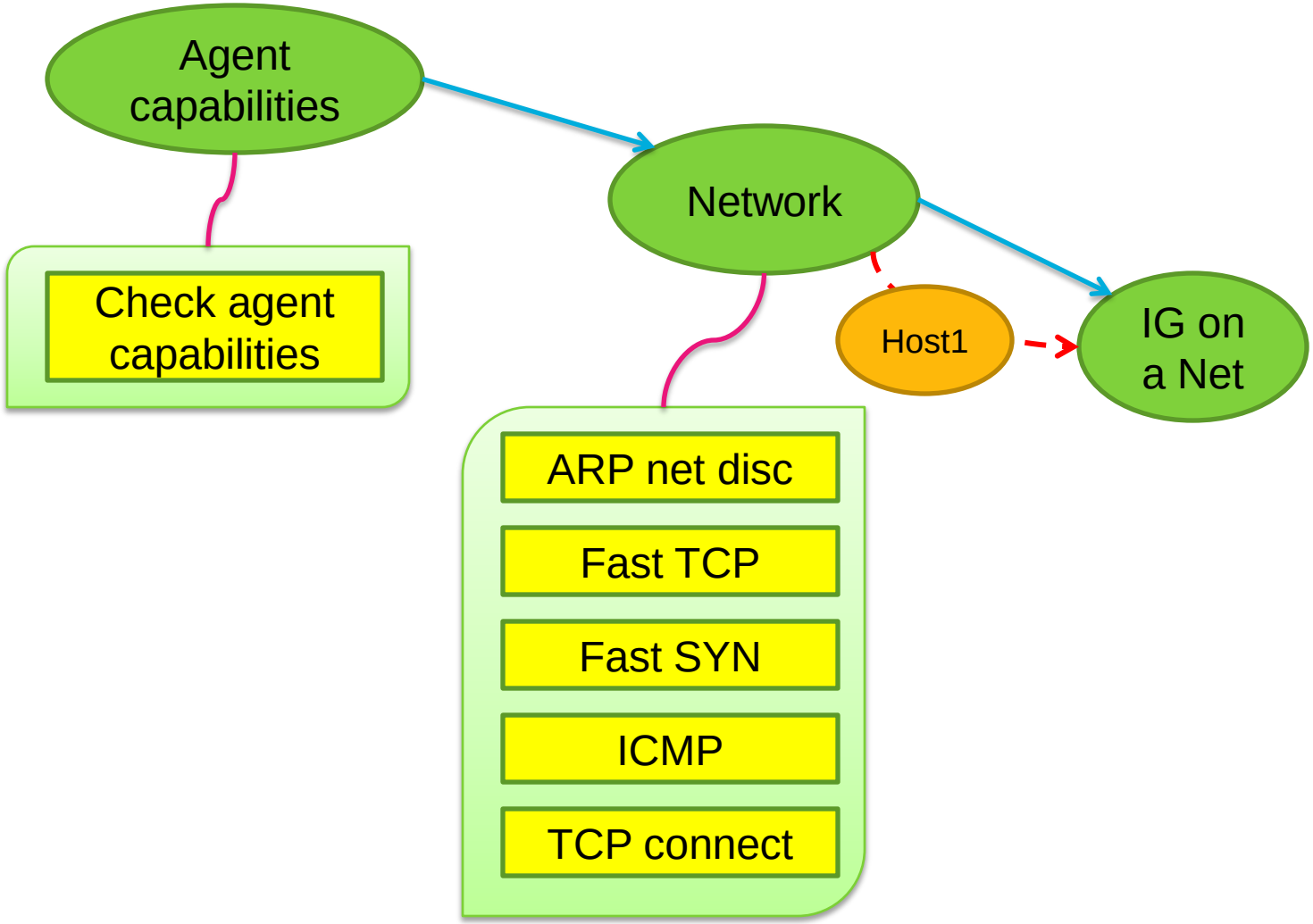
Planner

- The next component is the Planner, whose main tasks are:
  - Keep track of the final goal status
  - Backtrack its dependencies
  - Ask the Strategy the actions to execute to solve any Goal
  - When there are no possible actions to resolve a given goal, set it to Invalid
  - Re analyze the graph when there's new information (an action finished or produced something)

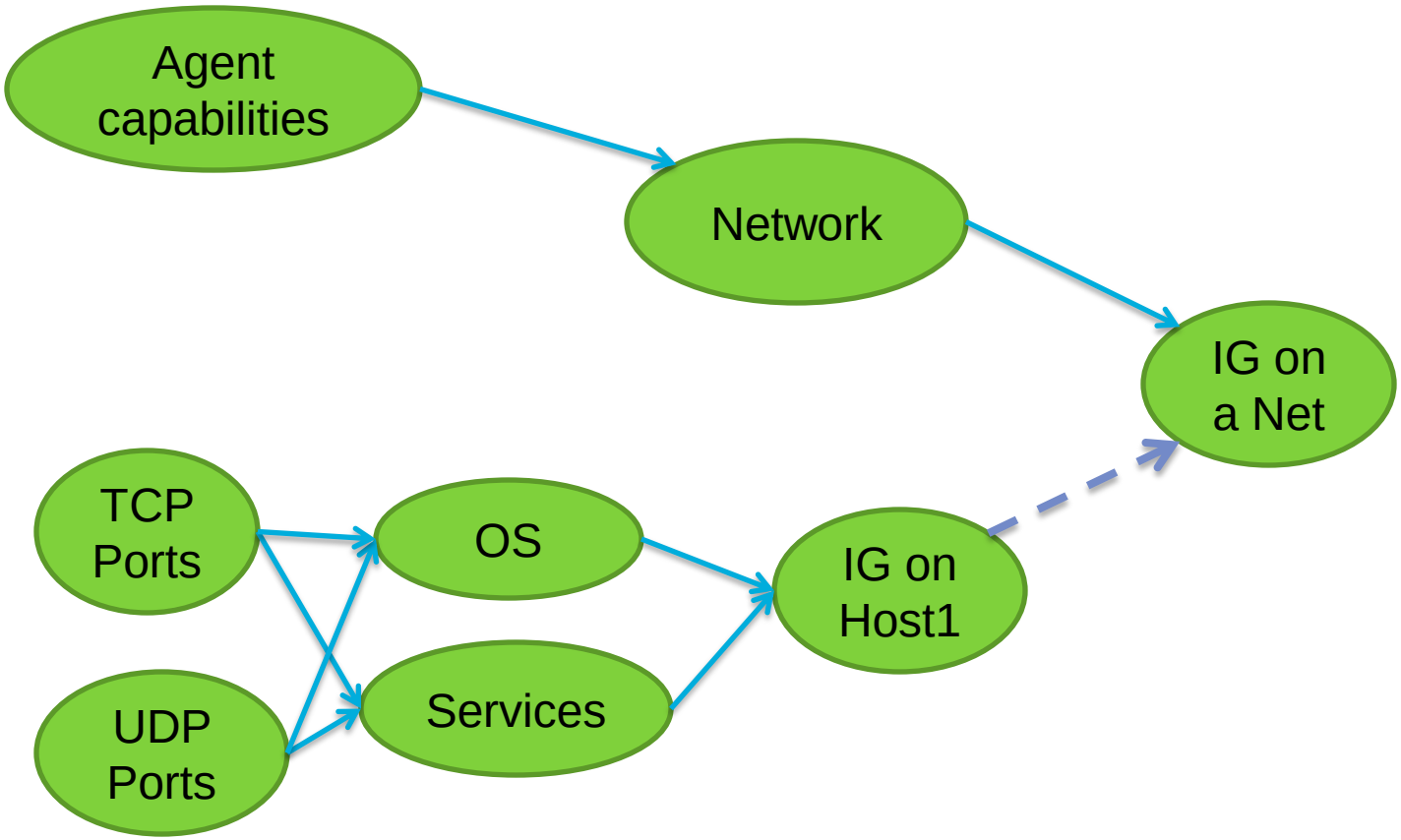
Dynamic graph

- Faced with a partially known network we need to construct the graph dynamically
- The simpler scenario is to gather information from a network, we'll have to represent:
  - a network
  - a dynamically found set of hosts
  - the information gathered for each host
- In more complex scenarios we have to deal with the Agents produced during the attack

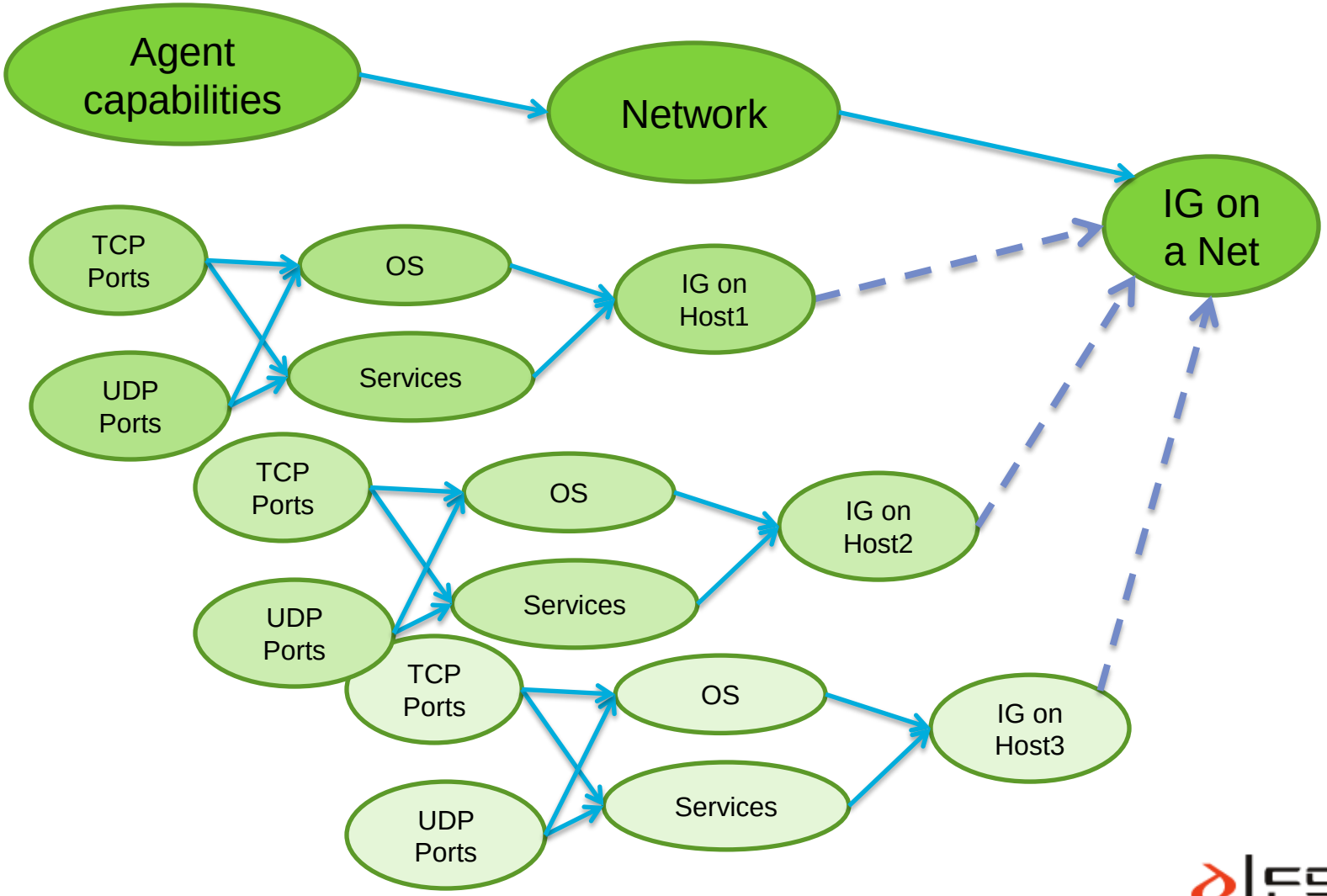
Information Gathering



New goal added



Graph grows on runtime



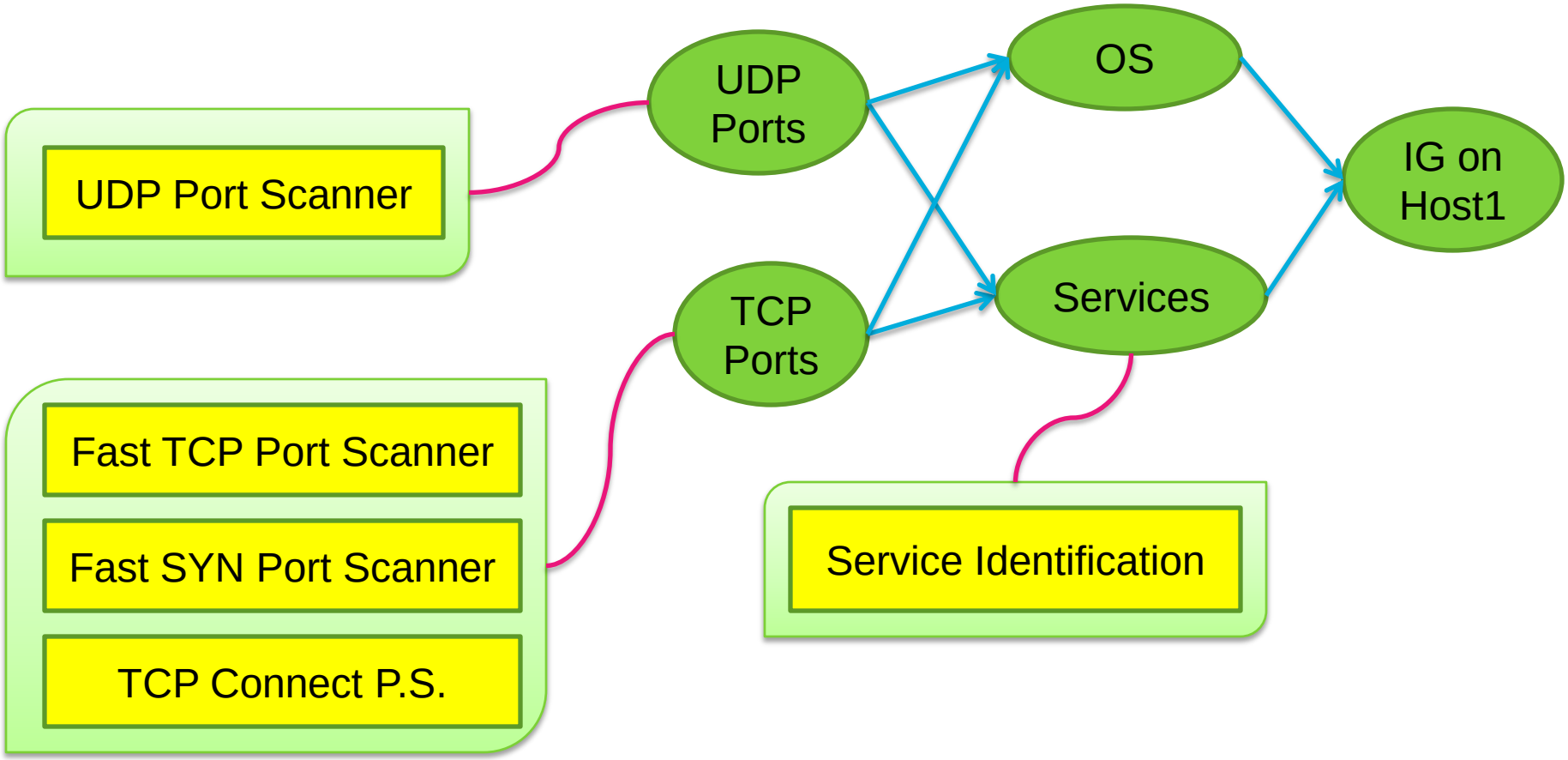
Online planning

- We need Online planning
  - deal with initial uncertainty
  - during an attack, more information is gained about the targets (details of OS, applications, configuration)
  - can be incorporated and modifies the graph
- Information Gathering is part of the attack
  - IDS should detect an attack during the IG phase
- There is not only one path to the objective
  - react to network changes for more accurate representation of attacks

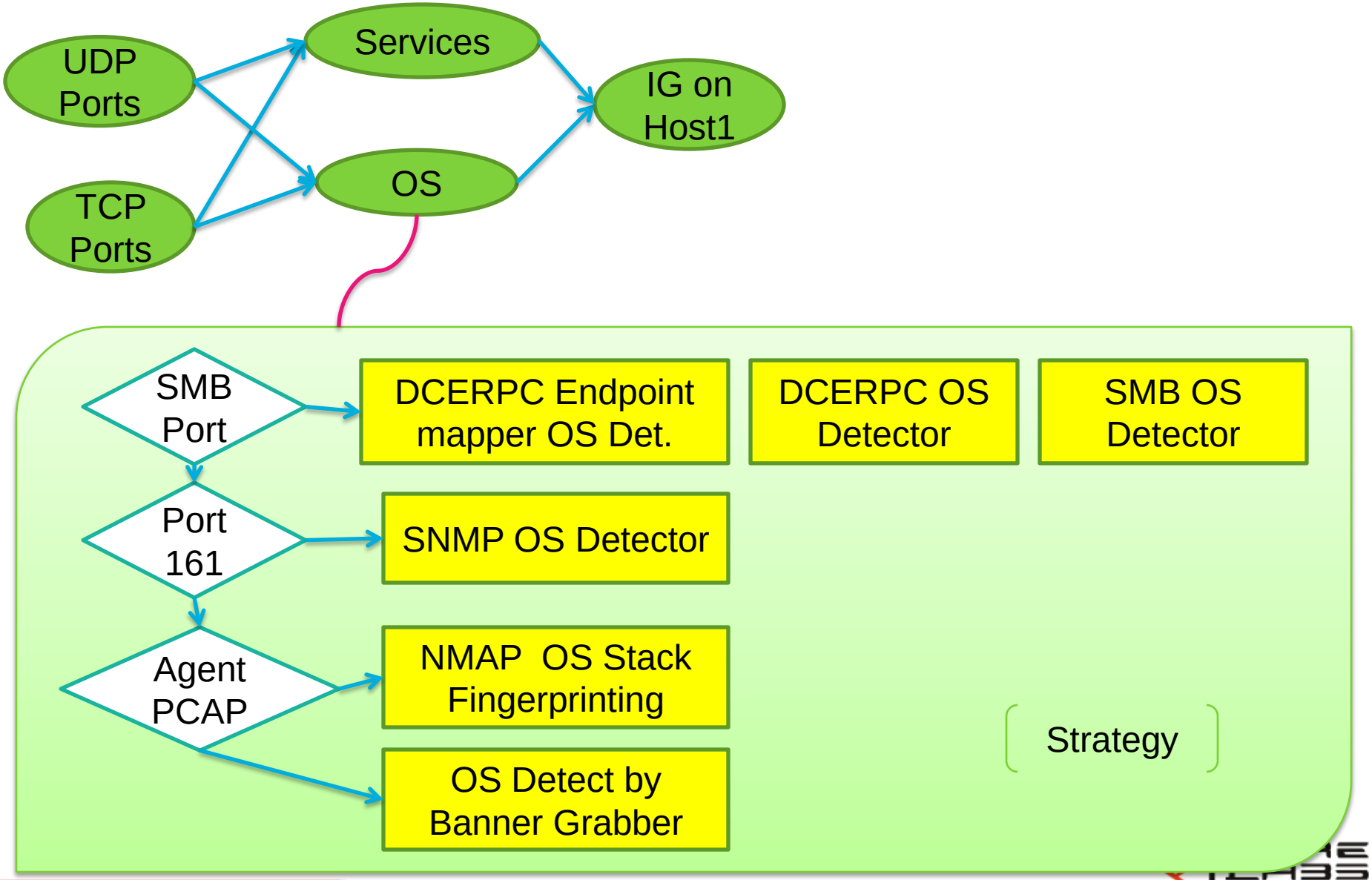
Environment knowledge

- The results of all the actions are collected in a database
  - attacker's representation of the network
  - also called "state of belief" in AI
- Necessary in a partially observable and non-deterministic domain
  - another difference with classical Attack Graphs
  - this is not a perfect information game!

Continuing the IG example

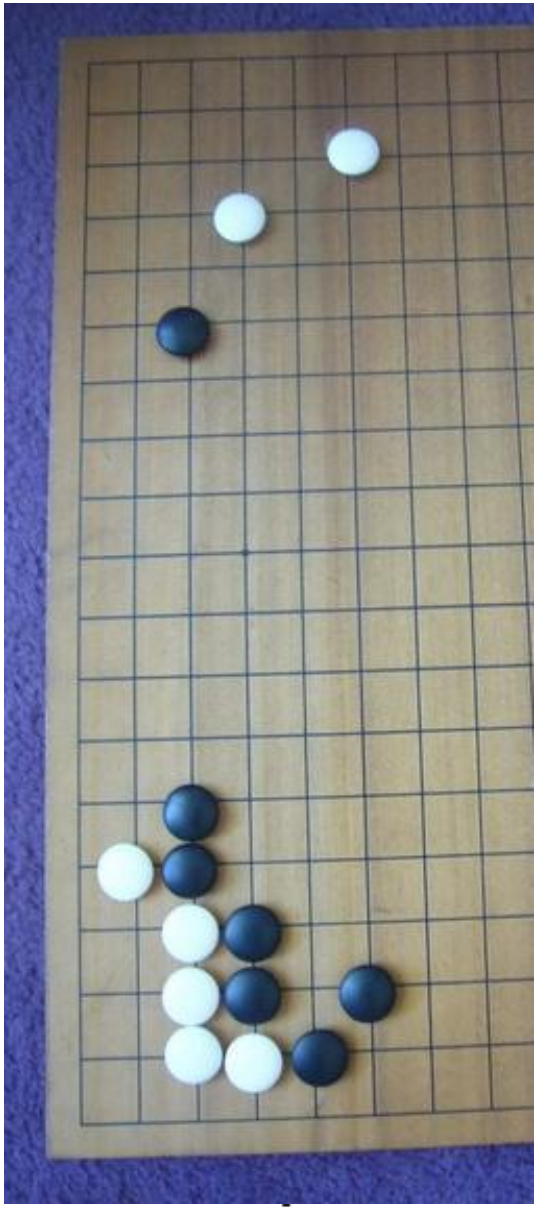


OS Detect Strategy

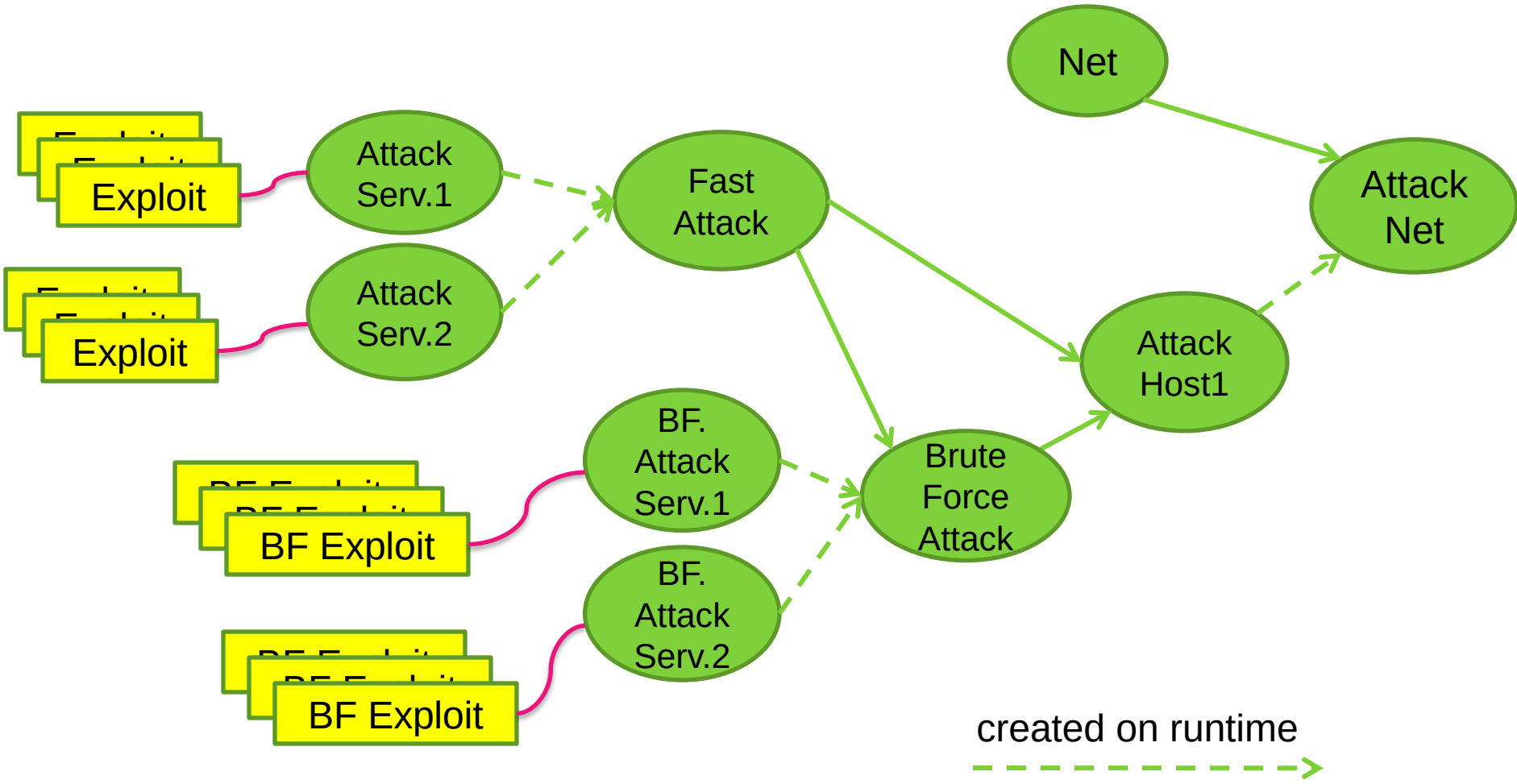


Improve Scalability

- HTN (Hierarchical Tasks Network) is a way to define compound tasks as set of simpler tasks
  - e.g. the OS Detect Strategy
  - can be compared to the Josekis in Go (opening books)
  - strategies have local resolution
  - suppose independence from the rest of the plan
- This is a way to incorporate expert knowledge



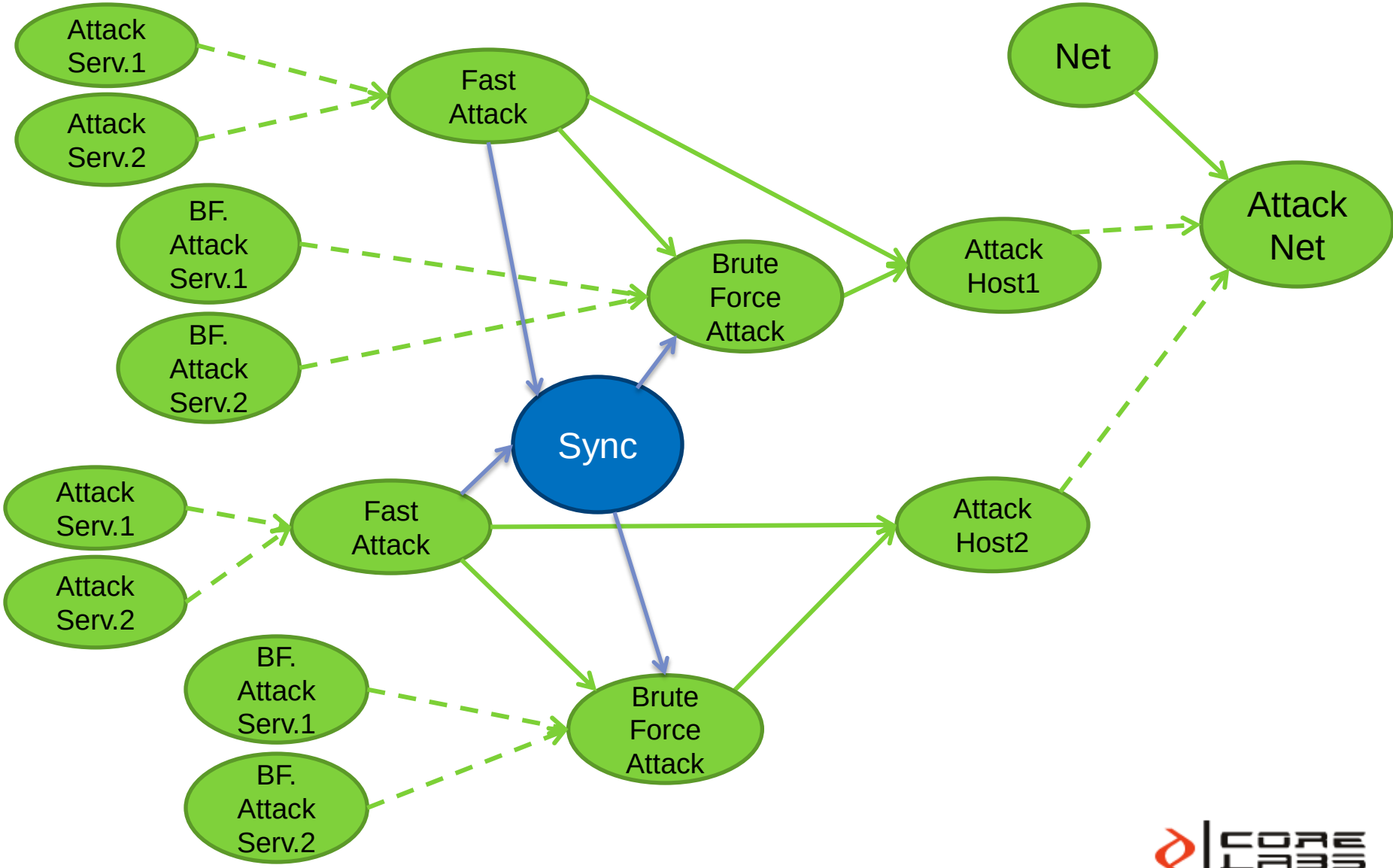
Attack & Penetrate



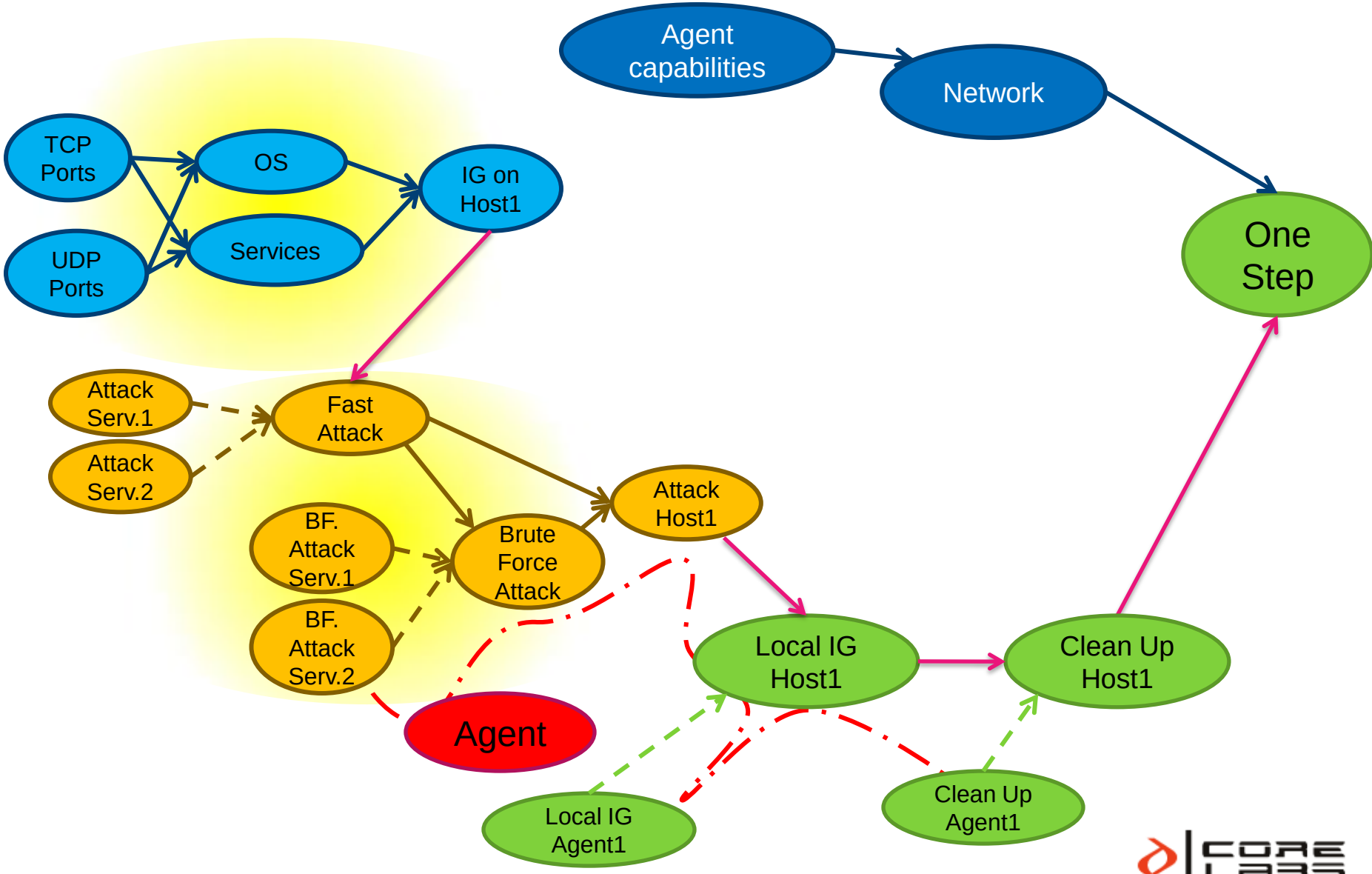
Parallelization – Scheduler

- Exploit execution takes time
  - attack vector latency (network / client-side)
  - wait for time-outs (e.g. the user opens an email)
  - brute force memory addresses
- Exploits are launched in parallel
  - maximize the number of different running tasks
  - keep balance between scheduling same kind of tasks and exhausting available resources
    - be polite! don't exhaust target machine resources
    - also limited by tolerated noise level in the network

Attack & Penetrate Sync



One Step subgraph



Ongoing research

Another direction for scalability

- Built on the model presented by Gerardo Richarte (BlackHat 2003)
  - Modern intrusion practices
- Find the optimal solution to the Attack Planning problem is NP-hard
  - can be reduced to the *Minimum Weight Set Cover Problem* (won't demonstrate it here, don't worry ;-)
  - constructing the complete Attack Graph is too expensive for real world scenarios
- Idea: use better planning tools to explore the graph, in order to find acceptable solutions

Search heuristics

- Don't construct the complete graph
- Use an heuristic to explore the action space
- There are several variations of A* search to find good solutions
- A* search is based on exploring first the states *s* that minimize

$$f(s) = g(s) + h(s)$$

  - where g(*s*) is the cost of reaching state *s* and h(*s*) is an estimation of the cost from *s* to the goal.

Relaxed problem

- h(*s*) = estimated cost from *s* to goal
- An *acceptable heuristics* must fulfill
  - h(s) ≤ real cost from s to goal
- How do we compute h(*s*)?
- An interesting solution is to compute the cost of solving a relaxed version of the problem
  - e.g. ignore delete effects
  - this is the monotonic hypothesis used by Ingols - Lippmann

PDDL experiments

- We have modeled the planning problem in the PDDL language
  - Planning Domain Definition Language
  - successor of the STRIPS language
- Language designed for the International Planning Competition
  - use the winning algorithms to generate plans
  - obtained good results with FF (Fast Forward) by Jörg Hoffmann

Cost of the actions

- We consider a multidimensional cost of the actions:
  - execution time
  - noise / generated traffic
  - probability of success (depends on reliability and novelty in the case of exploits)
  - complexity of execution
- Problem definition can include constraints on those dimensions
- This improves the realism of the model, but also complicates the planning problem.

A* Search with numeric effects

- Luckily, there are good heuristics for the case of numeric effects
  - the relaxed problem is to ignore negative numeric effects
- The plan is generated to minimize a combination of these dimensions
  - we have used Metric-FF by Jörg Hoffmann to solve domains with numeric effects

Example of PDDL action

```
(:action MSRPC_DCOM_exploit
:parameters (?s - host ?t - host)

:precondition (and
    (compromised ?s)
    (TCP_connectivity ?s ?t port139)
    (has_application ?t MSRPC)
    (or (has_OS_version ?t WinNT) (has_OS_version ?t Win2000)
    (has_OS_version ?t Win2003) (has_OS_version ?t WinXP)) )

:effect (and
    (installed_agent ?t high_privileges)
    (increase (time) 32)
    (increase (noise) 210)
    (increase (uncertainty) 135) )
)
```

Sample execution

- 3 networks with 40 hosts each
- 84 different actions
- resulting plan with 46 steps (pivoting 6 times)

```
instantiating 9673 easy, 794842 hard action templates
reachability analysis, yielding 2148 facts and 12973 actions
creating final representation with 2100 relevant facts
computing LNF
building connectivity graph
searching, evaluating 3189 states
63.29 seconds total time
```

Improve realism

- Numerical functions give a finer description of the action results
  - execution values obtained from our testing networks
  - statistics are updated when the action is executed in a particular environment
- Different types of attackers can be modeled:
  - minimize the execution time
  - minimize the generated traffic (for more stealthiness)
  - prioritize client-side

Conclusion

Summary

- Attack planning – from the attacker's point of view
  - consider all the steps of an attack, not only exploits
  - model the attacker's knowledge of the world
- Online planning
  - dynamically modify planning graph based on the results of the actions
- Scalability is a big challenge
  - custom HTN approach for efficiency (strategies)
  - parallelization
  - use state-of-the-art planning algorithms

Theory and Practice

- In modeling the Attack Planning problem, we need a balance between theory and practice
- A good model has a theoretical value
  - gives a better understanding of computer network attacks
- The level of detail and realism of the model must allow an efficient resolution of the resulting problem
  - as planning tools evolve this limit is pushed further

Some ideas for the future

- (Semi) autonomous agents
- Central command vs. agent guerrilla
  - assign missions to the agents and let them resolve how to obtain their goal
- Regulate communications between agents
  - communication channel may be intermittent
  - agents pursue their objective even without communications
  - distribute work between agents

Questions?

Thank you!

corelabs.coresecurity.com

Carlos Sarraute – carlos@coresecurity.com
Alejandro David Weil – aweil@coresecurity.com