CALDERA:
an automated adversary emulation system

Emilio Panti

UNIVERSITÀ DI PISA

Introduction

- I am Emilio Panti, computer science student at the Università di Pisa (Italy).
- My github page: link
- This presentation is informative only and it has been devoloped in the framework of my gradutation thesis at Università di Pisa (Italy). My supervisor is professor Fabrizio Baiardi.
- Presentation outline:
  1. **CALDERA (version 2.3.2)** structure, its operation model and how it operates.
  2. Our extension to define the **WORM plugin** and integrate it in CALDERA.

PART ONE: CALDERA

- An automated adversary emulation system defined in the MITRE ATT&CK™ framework.
- Developed by the MITRE Corporation.
- An actively developed project (github page: mitre/caldera).
- Last stable version: 2.3.2
- Minimal installation time
- Highly and easily customizable
- Low overhead – run it on a laptop
- Multiplatform: Windows, Linux, OSX
- Requirements: Python 3.5.3+

CALDERA Philosophy

Adversary emulation "**post compromise**":

- the adversary already has an initial foothold on a network (or in one host)
- focus on actions after the initial compromise – in this way CALDERA exercises network defence areas that are commonly weak and untested

Because of "post compromise" it **neglects**:

- how an adversary "gets in".
- precursors such as
  - vulnerability scanning,
  - intelligence gathering,
  - spearphising.

How to emulate an adversary

- Knowledge base needed → matrix MITRE ATT&CK
- MITRE ATT&CK matrix:
  - A knowledge base of adversary **tactics** (subdivided into **techniques**) based on real-world observations
    - **12** tactics
    - ~ **250** techniques

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	Binary Padding	Bash History	Application Window Discovery	Application Deployment Software
External Remote Services	Command-Line Interface	Account Manipulation	AppCert DLLs	BITS Jobs	Brute Force	Browser Bookmark Discovery	Distributed Component Object Model
Hardware Additions	Compiled HTML File	AppCert DLLs	AppInit DLLs	Bypass User Account Control	Credential Dumping	Domain Trust Discovery	Exploitation of Remote Services
Replication Through Removable Media	Control Panel Items	AppInit DLLs	Application Shimming	Clear Command History	Credentials in Files	File and Directory Discovery	Logon Scripts
Spearphishing Attachment	Dynamic Data Exchange	Application Shimming	Bypass User Account Control	CMSTP	Credentials in Registry	Network Service Scanning	Pass the Hash
Spearphishing Link	Execution through API	Authentication Package	DLL Search Order Hijacking	Code Signing	Exploitation for Credential Access	Network Share Discovery	Pass the Ticket
Spearphishing via Service	Execution through Module Load	BITS Jobs	Dylib Hijacking	Compile After Delivery	Forced Authentication	Network Sniffing	Remote Desktop Protocol
Supply Chain Compromise	Exploitation for Client Execution	Bootkit	Exploitation for Privilege Escalation	Compiled HTML File	Hooking	Password Policy Discovery	Remote File Copy

a part of the ATT&CK matrix

What is an adversary?

MATRIX ATT&CK

- ADVERSARY = collection of **techniques**
- TECHNIQUE = specific **behavior** to achieve a goal and that may be a single step in a string of activities to complete the attacker's mission

MORE "ABSTRACT" CONCEPT

CALDERA

- ADVERSARY = collection of **abilities**
- ABILITY = specific ATT&CK technique **implementation** (procedure) stored in YML format

PRACTICAL / EXECUTABLE CONCEPT

Distinct CALDERA abilities can implement the same ATT&CK technique

Example of a technique description in MITRE ATT&CK

File and Directory Discovery

Adversaries may enumerate files and directories or may search in specific locations of a host or network share for certain information within a file system.

Windows

Example utilities used to obtain this information are `dir` and `tree`. [1] Custom tools may also be used to gather file and directory information and interact with the Windows API.

Mac and Linux

In Mac and Linux, this kind of discovery is accomplished with the `ls`, `find`, and `locate` commands.

ID: T1083

Tactic: Discovery

Platform: Linux, macOS, Windows

System Requirements: Some folders may require Administrator, SYSTEM or specific user depending on permission levels and access controls

Permissions Required: User, Administrator, SYSTEM

Data Sources: File monitoring, Process monitoring, Process command-line parameters

Version: 1.0

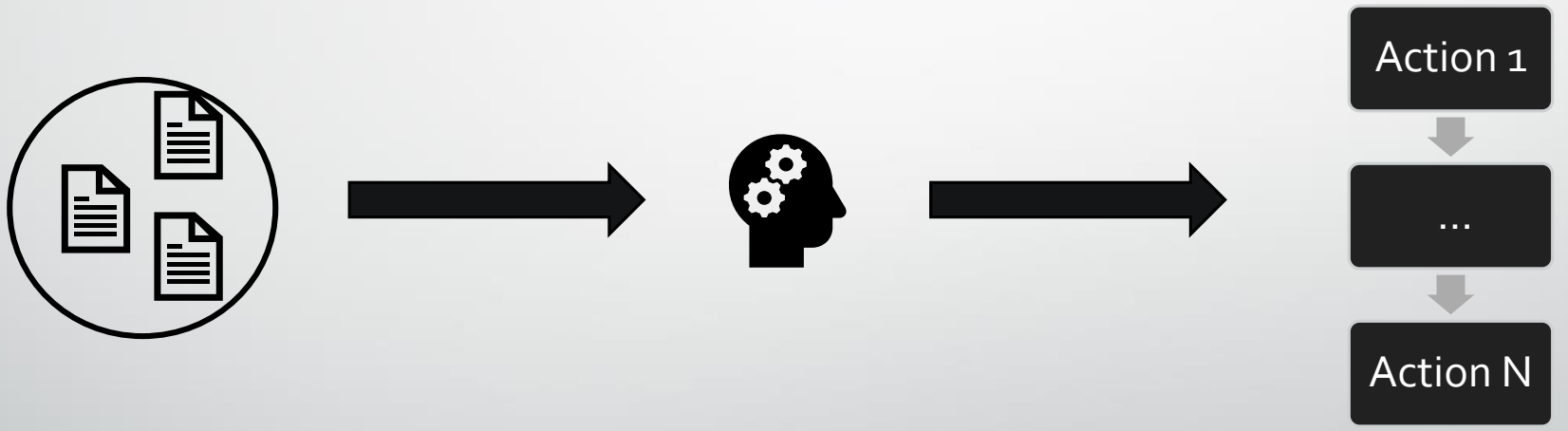
Mitigations

This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.

Example of a yml file describing a CALDERA ability ->

```yaml
- id: 0360ede1-3c28-48d3-a6ef-6e98f562c5af
  name: GetComputers (Alice)
  description: Get a list of all computers in a domain
  tactic: discovery
  technique:
    attack_id: T1003
    name: discovery
  platforms:
    windows:
      psh:
        command: |
          Import-Module .\PowerView.ps1 -Force;
          Get-NetComputer
        payload: PowerView.ps1
        parser:
          name: regex
          property: host.host.fqdn
          script: '[\S]+'
```

- CALDERA maps an adversary profile into a **chain** of actions.
- A **planner:**
  - **creates the chain** to simulate the adversary behavior
  - is a module to choose the abilities and their execution order in a running operation
  - CALDERA default planner -> **the *Sequential* planner**



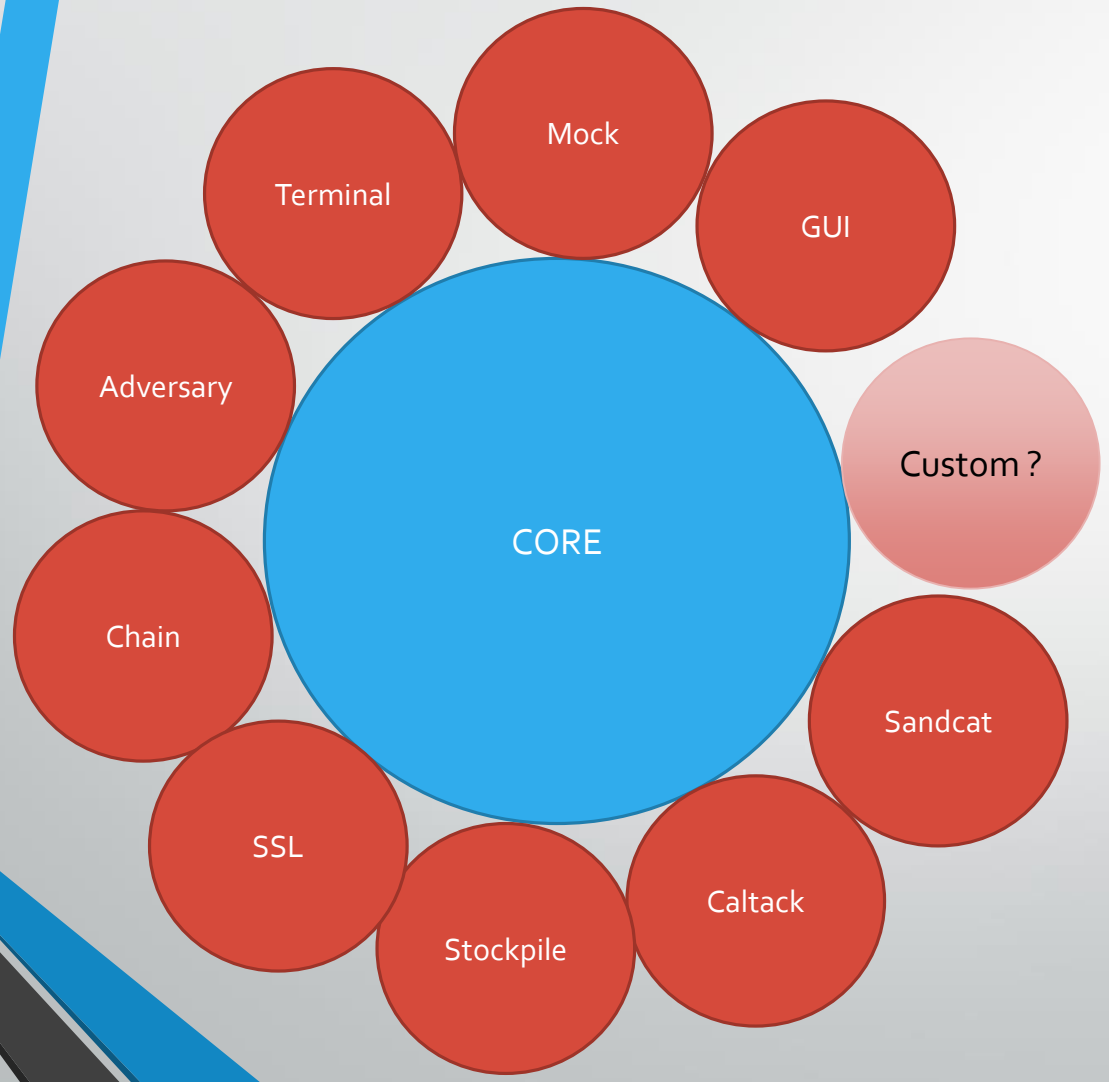
ADVERSARY (described as a collection of CALDERA abilities)

CALDERA's planner

CHAIN of executable actions to emulate the adversary

CALDERA Server:

- An open plugin architecture on top of a core.



- **Core**: the basic functionalities (database management, authentication, etc.) and services

- **Plugin**:
  - separate git repository that plugs new features into the core system.
  - It resides in the plugins directory and is loaded into CALDERA through a configuration file.

CALDERA: **Key concepts** (1)

- **Agent**: an individual computer running a RAT/agent that connects to CALDERA in order to get instructions, executes them and returns the results.
  - Trusted/untrusted agents: an agent becomes untrusted if it remains silent for more than a configurable time interval.
- **Group**: a collection of agents to support concurrent operations against multiple computers rather than sequential one.
- **Ability**: an implementation of a specific ATT&CK technique (procedure).
  - Currently supported platforms (and executors): Linux/Darwin (sh), Windows (psh/cmd/pwsh).
  - The ability command section may contain variables, identified as #{variable}, which can be filled with facts.
  - The ability output can be parsed into facts.

CALDERA: **Key concepts** (2)

- **Fact:** an identifiable piece of information (with a score) about a computer:
  - Host-fact: fact where the property major component = host (e.g. host.user.name). Operations with multiple agents can substitute the variable in the abilities only with the facts the specific agent has collected.
  - Global-facts: any fact that is not a host-fact. Used by all operation agents.
- **Link**: a (unique) variant of an ability where distinct fact combinations have replaced variables and are ready to be executed.
- **Adversary**: a threat profile contains a set of abilities, that may be grouped into phases to enable a user to choose their execution order.
- **Operation**: it starts when pointing an adversary at a group to run all capable abilities.

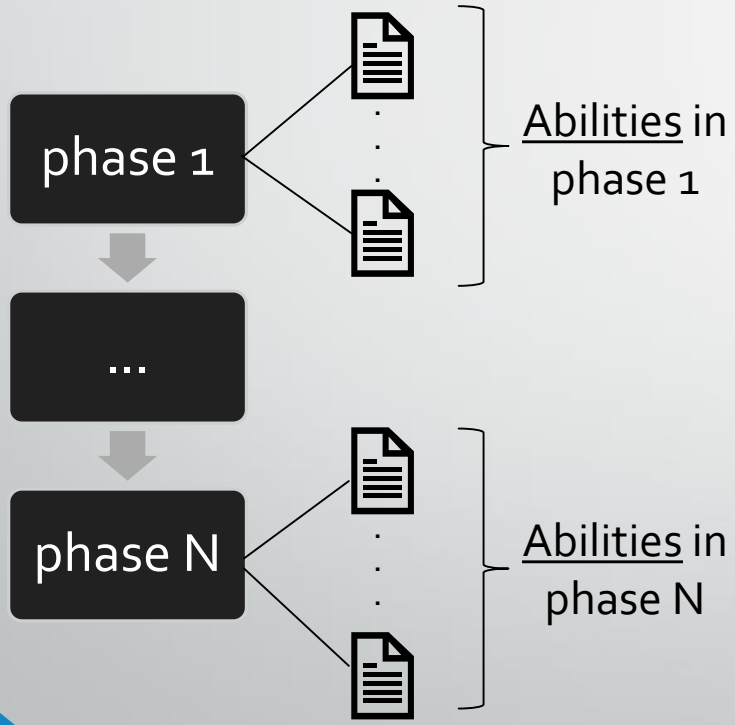
My proposals

1. **Trusted/untrusted agents**: this enables a running operation to choose whether to allow or not untrusted agents.
   - This prevents an entire operation from stopping as it waits for the results of a "dead" agent.
2. **Host-facts**, and the following distinction with respect to the global-facts, are the evolution of private-facts, a concept I have introduced and implemented.
   - Private facts: same concept of host-facts but defined by the key word "private" in the property field (e.g. host.file.private)
   - from version 2.3.1: private-facts -> host-facts

MITRE Corporation has accepted both proposals and incorporated in its implementation.

"Ingredients" of an operation:

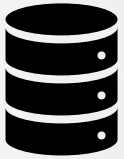
ADVERSARY



GROUP

Collection of agents partecipating in the operation

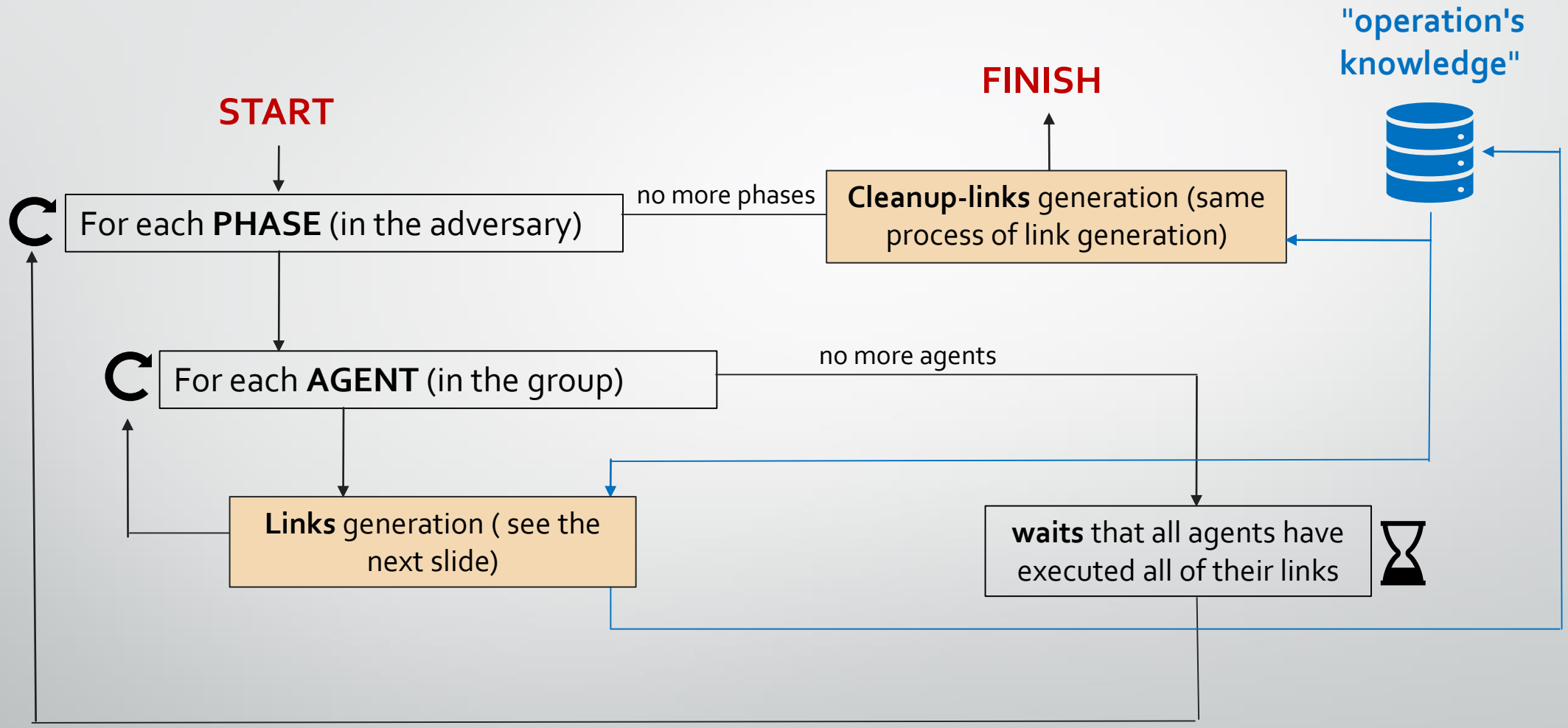
"OPERATION's KNOWLEDGE"



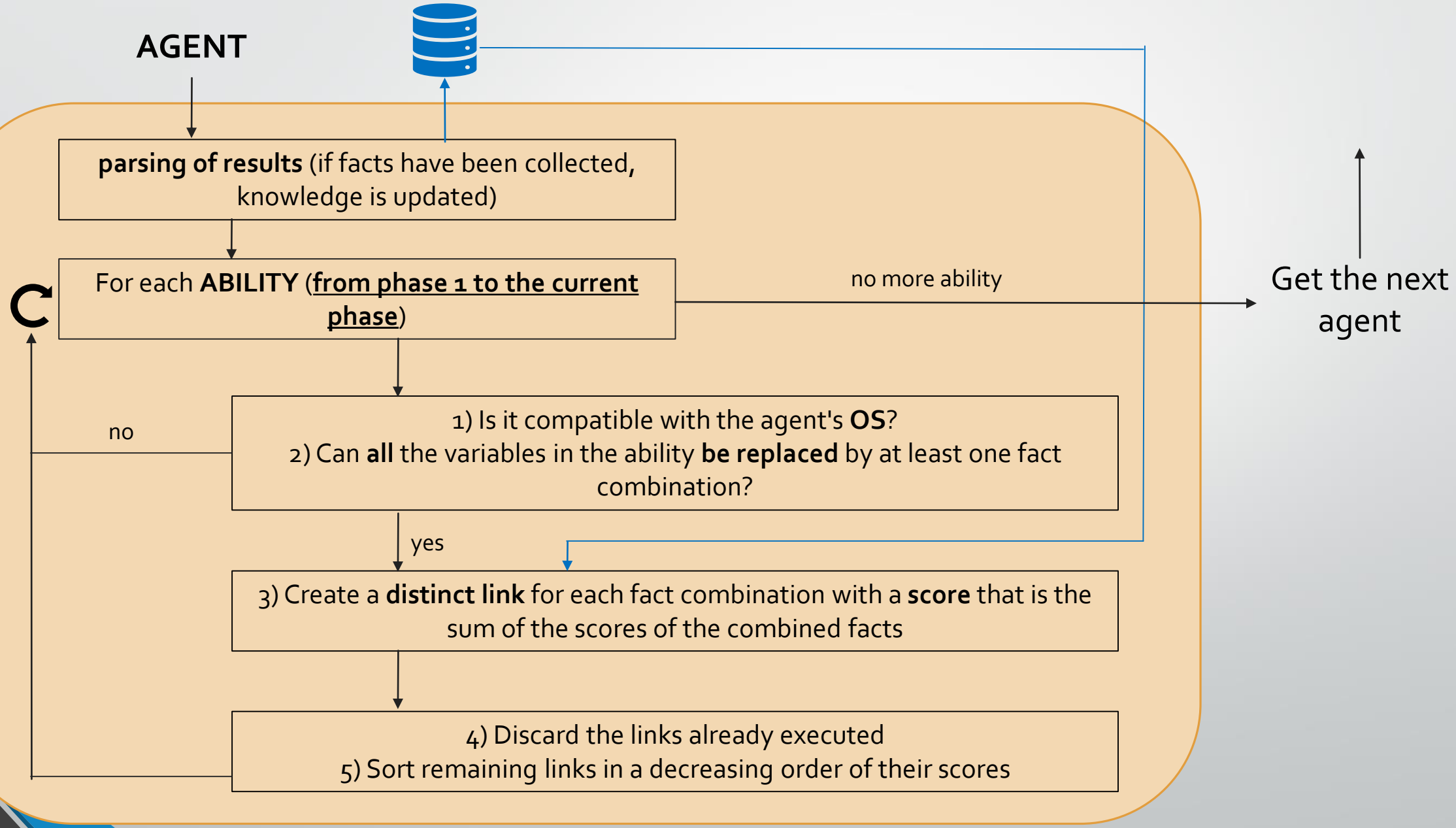
Collection of all collected facts, both host-facts and global-facts.

Before starting an operation, some optional configurations may provide a "pre-knowledge" by attaching a source of facts.

Life cycle of an operation (using the default sequential planner):



Links generation:



Worm: some definitions

- **Computer-worm** (or simply worm): an autonomous malware program that replicates itself to spread to other computers.

- **Worm** in CALDERA: operation with an adversary profile that aims to **automatically** expand and replicate the CALDERA agent in other computers by lateral movements.

- **Father agent:** agent that manages to expand/replicate to other hosts.
- **Child agents:** the copies of itself a father agent generates.

Modeling worm in CALDERA: problems

- New agents joining the operation after successful lateral movements:
  - may not execute the attack phases in the expected order - due to the forced recovery of the phases already executed (by the father-agents) as if they were a single phase.
  - may never start the operation - if they join in the last phase,
  - may not execute some abilities - due to the possible dependencies between them.
- The new global-facts the new agents collect during the recovery of the phases could force all the agents in the operation to repeat some abilities resulting in violation of the expected execution order.
- Complex to deduce father/child relationships.
- Impossible to define a worm goal (eg an agent that satisfies some conditions).

- **LOCATE PROBLEMS** most problems arise because of two implementation choices:
  1. the operations are mainly driven in the first place by the execution phase and then by the agents
  2. for each agent, the links of all the phases executed up to now are generated -- not just the current one

- **SOLUTIONS**:
  1. **worm-operation :** a new operation type with a distinct logic.
  2. **worm-planners:** a new class of planners that generate links for the current phase of the attack only.

WORM PLUGIN

PART TWO: WORM PLUGIN

- Github page: WORM plugin.
- Plugin goals:
  1. Create a system that solve CALDERA problems to **automatically** models worm;.
  2. Generate an **agents map** to correlate fathers and child agents.
  3. Define a worm **goal** and a worm **policy**:
     - stop the attack as soon as a goal is reached.
     - continue the attack until expansion is possible.
  4. Return a more detailed **report** of a worm attack.
  5. Supports a simple and fast way to **split the agents** in two groups:
     - those that have fulfilled the attack goal
     - the other ones

To be noticed

- To work properly, this plugin needs some extra features with respect to the basic CALDERA version.
- You can download here the version with these features.
- The two versions differ because of a further agent property: the **father**.

- The father parameter makes it possible to generate an **agents map** and correlate fathers with child agents.
- Why change the CALDERA code rather than the WORM plugin?
    1. Easier to implement and clearer -- only 10 lines of code.
    2. It may be useful even for "normal" CALDERA operations -- therefore this change could be accepted and integrated (two open pull request PR/Caldera and PR/Sandcat)

Goal

- A formula in **Conjunctive Normal Form (CNF):** a conjunction of clauses, where the clauses are a disjunction of literals. Every literal can be a condition:
  1. on the **properties** (namely the fields of the db *core_agent* table) **of an agent.**
  2. on **host-facts** collected by an agent.

  Example: (host.file.sensitive='fileX') ∧ (platform='linux' ∨ platform='windows')

- The goal definition is closely related to the **adversary profile** we would like to execute as it defines the host-facts the agents can collect.
- A goal is **achieved** if in ANY clause at least ONE condition is satisfied.
- **Goal-agent:** an agent that satisfies the goal.

Worm-planners

- New class of planners.
- Worm-planner: a module to decide the abilities a running worm-operation should use and their execution order .
- Worm-operation planners have to implement two functions:
  1. **create_links()**: given a worm-operation and an agent, it generates all possible links for the next phase the agent should execute.
  2. **create_cleanup_links()**: given a worm-operation and an agent, it generates the cleanup links for the phases the agent has executed.
- Default worm-planner: **the *worm_sequential* planner:**
  - as its counterpart for normal operations, it orders generated links according in descending score order.

Worm-operation: basic logic

- New operation type with a distinct logic.
- This operations type is **driven by the agents** in the first place and then by the execution phase.
- This implies the agents execute the adversary profile **independently** (asynchronously) of each other.

  e.g. agent X can be in phase 1 while agent Y is in phase 4.
- Agent independence:
  1. speeds up both the entire attack and the expansion through lateral movements
  2. guarantees that all agents execute the **ENTIRE** adversary profile and the phases in the **EXPECTED ORDER**

Worm-operation: agent map & goal

During a running worm-operation, the GUI can build a view of:

- **"agents family tree"**: minimal reconstruction of the father / child relationships of the agents
- **"orphan agents"**: list of agents for which no father agent was found

Optional parameters added (with respect to normal CALDERA operations):

- **goal**: it is possible to set a goal for worm-operations
- **goal-policy**: if a goal is set, it is possible to choose one of the following :
  1. **first goal-agents**: the worm-operation ends as soon as any agent satisfies the goal or when no expansion is possible and any agent has concluded the attack.
  2. **maximum expansion**: the worm-operation will end **only** when no expansion is possible and any agent has concluded the attack.

  Note: Both policies may produce distinct goal-agents.

WORM plugin: new features (summary)

1. **Worm-operation:** a new type of operation.
2. **Worm-planners:** a new class of planners.
3. **Goal:** a new concept.
4. A **new report** for worm-operations with additional information regarding: goal, worm-policy and percentage of goal-agents.
5. Some further features for **agents**. After the end of a worm-operation, it is possible to:
   - compare the results of a finished worm-operation with other goals - associated with the executed adversary profile
   - split the participating agents into the goal-agents and the no-goal-agents

It is possible to use these features both via REST API and GUI.

Concluding remarks & possible integration

- All set plugin goals are met.
- The worm-operations have a clearer and more confined behavior than normal operations. All phases are performed in the expected order and for all participating agents.
- The WORM plugin could be completely integrated with CALDERA by:
  1. introducing the **father parameter** for agents (PR)
  2. extending the concept of **goals** also to normal operations
  3. adding the **agent-map** section even to normal operations
  4. Enabling the choice **of one of two distinct logics when running an adversary profiles**:
     - assign a larger priority to the phases than to the agents --> normal CALDERA operations logic
     - assign a larger priority to the agents than to the phases--> worm-operations logic