Strategies based on Att&ck Matrix

The gray agent can

- *Reboot a host.* This will flush all active credentials within the host; if the attacker were to dump credentials immediately after a reboot, it would not receive any.
- *Log a user into a host.* This will store the user credentials in the host memory. Only accounts that are authorized to log in and are not currently logged in may log in to a given host.
- *Open a network connection between two hosts.* This simulates two hosts talking to each other on the network. Only hosts are authorized to share network traffic can open connections to each other.
- *Close a network connection between two hosts.* If there is an active connection, it will be terminated
- The gray agent executes a batch of multiple actions during its turn, where each capability has a predefined probability, varied during testing, to execute during an action turn within a batch

Strategies based on Att&ck Matrix

An attacker is composed of three components:

- *Capabilities.* the things the adversary can do during its turn;
- S*trategy*: how the adversary chooses a capability to execute
- *Knowledge*: what the attacker can know about a given environment and it is described as a set of predicates. Possible predicates are

knowsCreds(*A*)	The attacker knows the credentials for account *A*.
knowsConnected(*X,Y*)	The attacker knows that hosts *X* and *Y* are connected.
knowsRemote(*A,X*)	The attacker knows that account *A* is authorized to remotely log in to host *X*.
knowsAdmin(*A,X*)	The attacker knows that account *A* has administrative access to host *X*.
knowsActiveConnection(*X,Y*)	The attacker knows there is an active connection between hosts *X* and *Y* .

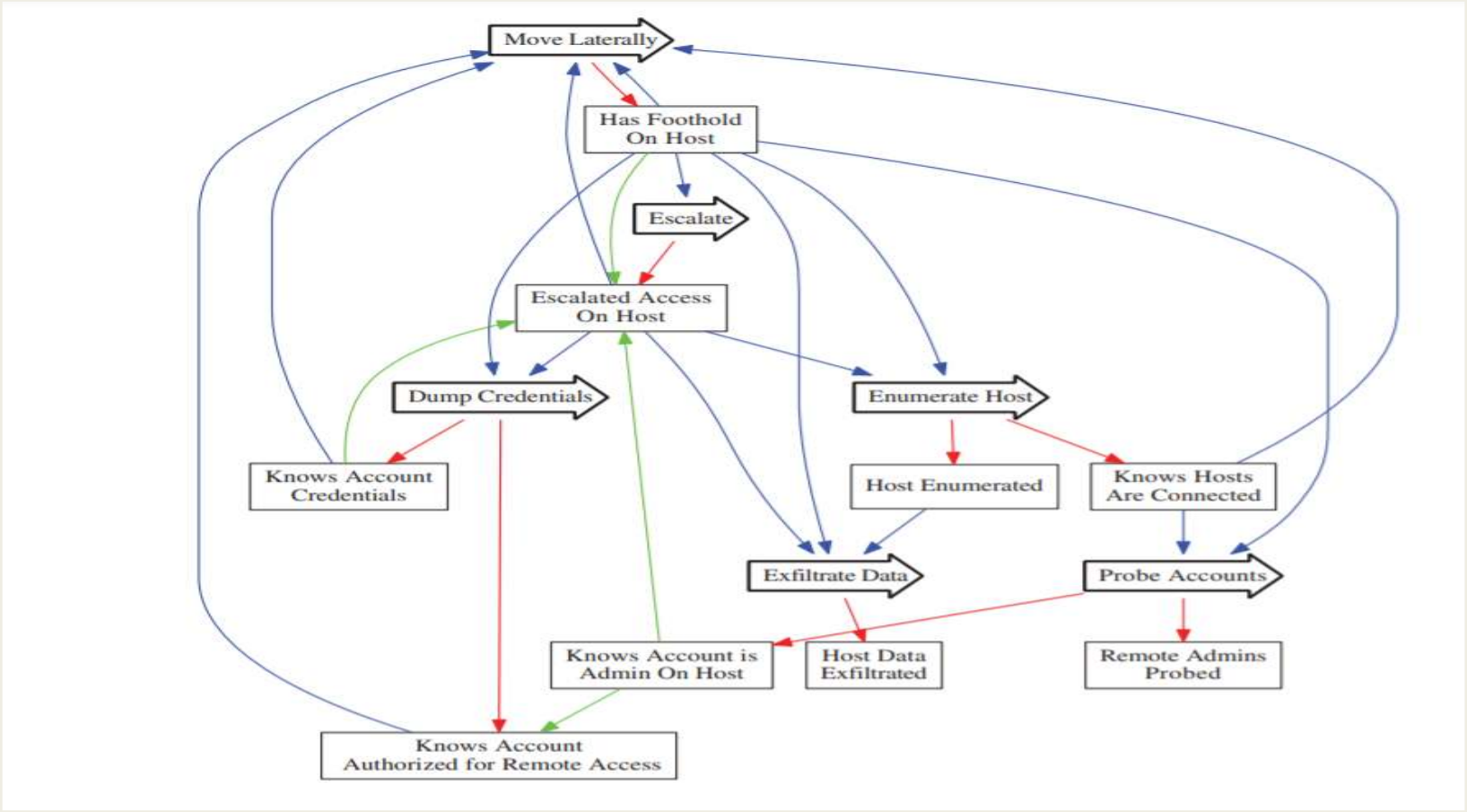
Strategies based on Att&ck Matrix

The attacker capabilities describe the things it can do during its turn; currently six tactics from the Att&ck matrix

1. **lateral(X, A)** Laterally move to host *X* using account *A* (*Lateral Movement*). The attacker has a foothold on Y and not on X, knows *X* and *Y* are connected, the credentials to account *A*, have an escalated foothold on *Y* , *A* is authorized for remote login on *X*, After, it will have a foothold on *X*.
2. **dumpCreds(X)** Dump the active credentials on host *X* (*Credential Access*). The attacker has an escalated foothold on *X*. After, the attacker will know the credentials of any users who stored on the host.
3. **probeAccounts(X)** Find host X admins (*Discovery*). The attacker has a foothold on *Y* connected to *X* and not a foothold or probed accounts on *X*. After, it will know all admin accounts on X, both local and domain.
4. **enumerateHost(X)** Enumerate host *X* (*Discovery*, *Collection*). The adversary has an escalated foothold on *X* and has not previously. enumerated X. After it will know all hosts *X* is connected to and active connections to and from *X*.
5. **exfltrate(X)** Exfltrate sensitive data from host *X* (*Exfltration*). The attacker has enumerated and an escalated foothold on *X* and not previously exfltrated from *X*. The adversary successfully exfiltrates once all relevant files.
6. **runExploit(X, E)** Run exploit *E* against host *X* (*Lateral Movement*). The adversary has a foothold on *Y no foothold on X* , knows *X* and *Y* are connected, and has not tried to exploit *X* using *E* in the past. After it will have a foothold on *X iff X* is vulnerable to *E*. The attacker only learns whether a host is vulnerable to E by trying to exploit it
7. **escalate(X)** Escalate privileges on host *X* (*Privilege Escalation*). The adversary must have a low-privilege foothold on *X*. After the adversary has an escalated foothold on *X*. A lateral movement with *lateral*(*X,A*) can skip escalation on *X* if *A* is an administrator.
8. **cautious actions** An attacker strategy can choose to execute to *runExploit*(*X,E*) and *lateral*(*X,A*) *cautiously*; that is, without the risk of being detected if the attacker enumerates the launching host fIrst and chooses a target with which the launcher already has an active connection.

Strategies based on Att&ck Matrix

The attacker capabilities describe the things it can do during its turn



Strategies based on Att&ck Matrix

The attacker S*trategy* chooses the capability to execute.

The paper considers two kind of strategies

- Immediate execution strategies : they only consider the current execution environment and do not attempt to plan for the future
  - *Random*
  - *Greedy:* selects among the possible capabilities based on a pre-confgured profile: exfltration, enumeration, lateral movement, remotely probing accounts. If any is possible, it executes it, prioritizing according to its list position. Otherwise, it randomly selects another action
  - *Finite state machine:* it repeats a fixed sequence: escalate, dump credentials, enumerate, exfiltrate, probe accounts, and then lateral movement
  - *Cautious strategy:* only greedy and FSM that prefer cautious lateral movements and exploit

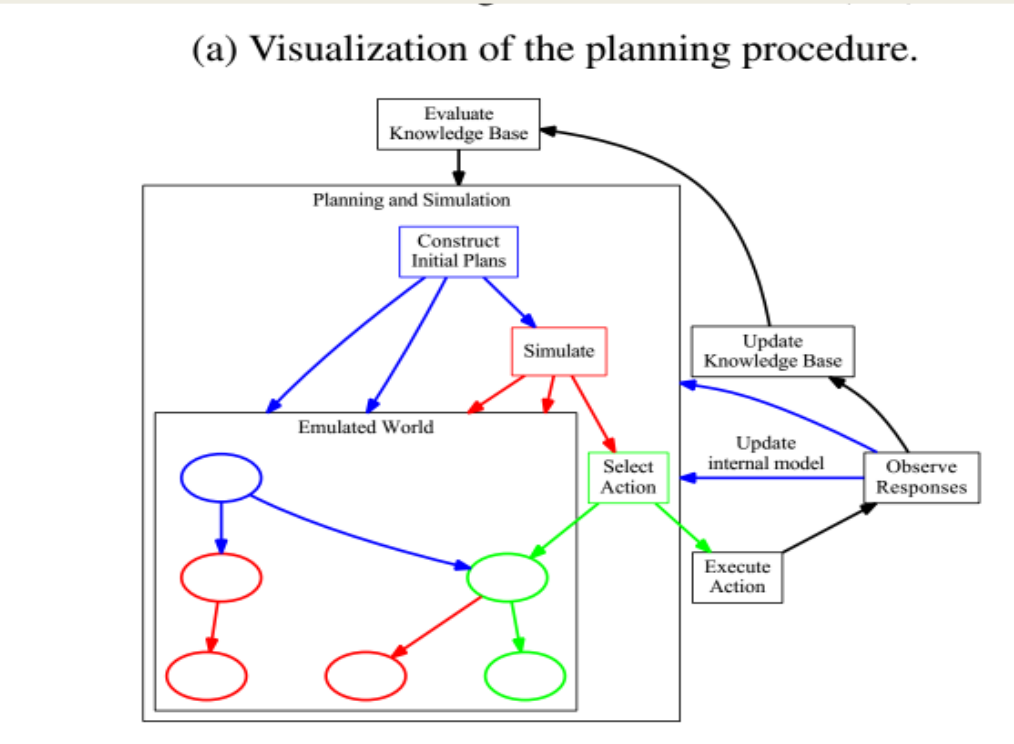
Strategies based on Att&ck Matrix

The attacker *Strategy* chooses the capability to execute.

- Planning based

  To combat uncertainty and prioritize actions, the planning agents all feature an *uncertainty handler* and a *prioritization procedure*

  1. Evaluate currently available knowledge and identify immediate actions.
  2. Extrapolate what may exist in the world using a *simulation* procedure.
  3. Construct a set of plans over the simulated world, each evaluated by the prioritization procedure.
  4. Execute the action represented by the most prioritized plan.
  5. Observe responses, updating the internal knowledge base and re-running step 1.



Multiple Plan and multiple simulation the most selected action is executed

Strategies based on Att&ck Matrix

Plan Priorization Procedure

1. Assign each action an individual *action score*.
2. For each action in the plan sum up the action scores, dividing each action score by its position in the plan.
3. To mitigate plan length differences, divide each plan score by its length

Action	Score	Action	Score
dumpCreds	1	*runExploit*	1
lateral	4	*probeAccounts*	2
enumerateHost	5	*exfiltrate*	100
escalate	1		

Plan	Score
dumpCreds; lateral; enumerate; exfil	7.42
dumpCreds; probe; lateral; enumerate; exfil	4.92
probe; dumpCreds; lateral; enumerate; exfil	5.02
runExploit; escalate; enumerate; exfil	7.04

Strategies based on Att&ck Matrix

Uncertainty handler

- Generate several worlds that matches the current attacker knowledge base
- In each world plans are builted and ranked and an action is selected
- The most selected action is executed

Full planner

The planner initializes an entirely new world – independent of the real one – combines that world with its current knowledge base to ensure consistency, and then runs a gray agent for some time

Lite Planner

An optimistic and smaller scale simulation of the full one that builds a smaller simulated world

1. With low probability, hosts that have not been exploited are vulnerable.
2. Any known host that has not been enumerated is connected with low probability, to unknown hosts
3. If the admins of a host are unknown, there are at least two admins, one known and one unknown.
4. Administrator credentials have high probability of being active on a host

Strategies based on Att&ck Matrix

Perfect strategy

The last strategy is the perfect knowledge strategy. A planner that has a complete and detailed knowledge of the target system. It is used as a comparison to evaluate all the other strategies

The defender

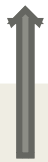
- The attacker can create a new network connection if it executes a lateral movement action from one host to another one where a connection does not already exist.
- Based on a pre-confgured probability, the defender will analyze each of these new connections; if the defender analyzes a new connection created by the attacker through a lateral movement, then the attacker's foothold on both hosts will be removed.
- If the attacker instead leverages an existing active connection between hosts, then the action is immune from detection.

Strategies based on Att&ck Matrix

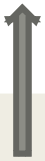
Some results

Average number of exfiltration (easy target)

	Finite	Random	Greedy	Lite	Full	Perfect	Finite	Random	Greedy	Lite	Full	Perfect
Min	4.7	0	4.7	4.7	4.7	4.7	4.7	0	4.7	4.7	4.7	4.7
Max	41.6	25	40	50	50	50	41.6	27.2	44.4	50	50	50
Median	14.2	7.1	13.3	17.6	15.7	23	22.2	7.1	15.3	22.2	20	31.5
Mean	16.1	7.7	15.2	19.5	18.5	24.4	21.5	8.1	17.2	23.5	22.0	29.9
Std. Dev.	8.8	4.6	8.0	10.9	10.4	13.7	9.9	4.7	9.0	12.2	11.8	14.7
Count	789	788	789	788	789	788	781	776	778	777	778	778



Not cautious



Cautious

Strategies based on Att&ck Matrix

Non cautious results

- On average the random agent performed the worst and the perfect knowledge the best.
- The performance range of the other four was small; the greedy agent is the worst with only 15.2%, while the lite planner is the best with 19.3%.
- The full planner performed slightly worse for exfltrations than the lite planner that almost doubled the credentials of the full planner.
- The FSM agent performed worse than the full planner but it obtained the most credentials.

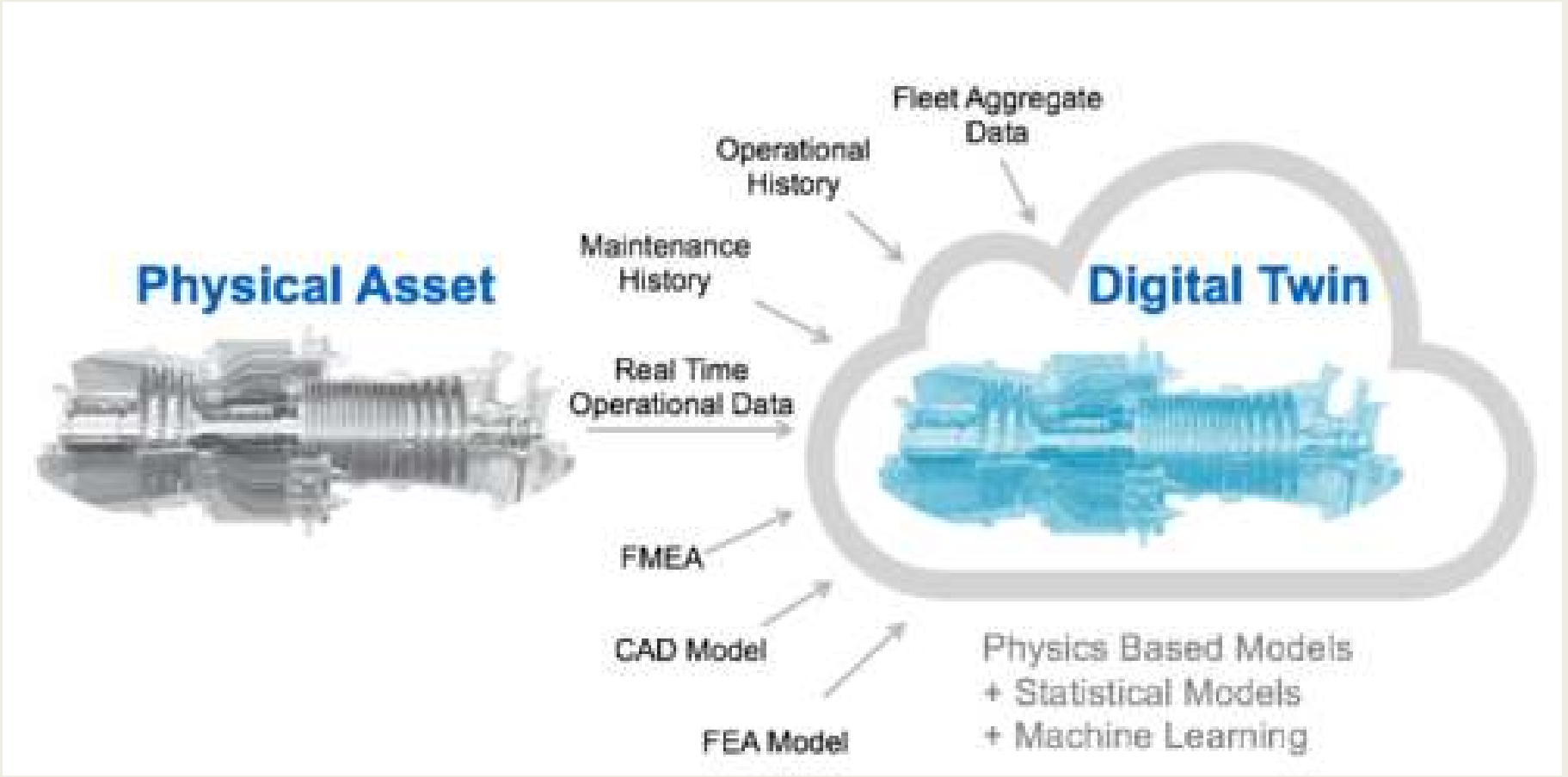
Cautious results

- all agents, aside from random, perform better than their regular counterparts. This was in part expected; the cautious configuration did not impose any additional cost even if it may require more turns for the adversary, due to the local host enumeration.
- The random agent performs the worst, the perfect knowledge one the best, and the lite planner performs second best.
- A much better performance of the FSM agent – not only does it exfltrate just about as much as the full planner, it also has three times the number of credentials.
- The greedy agent performs worse than all of the others aside from the random agent.

Simulation Environment

- Notice that the intrusions did not target the real system
- The noise generated by a simulated attacks prevents running intrusions with alternative strategies
- To evaluate and rank alternative strategies we cannot target the real system and use a digital representation of the target
- This digital representation is the digital twin of the real system
- Digital twin are becoming more and more popular to optimize mantainance and to predict when mantainance is needed (predictive maintance) for mechanical devices and robot
- Sensors on the device/robot collect information on the actual status that is transmitted to the digital twin and compute the maintainance planning
- The previous MITRE works shows one of the first examples that adopt a digital twin even for ICT security

Digital Twin in Industry 4.0



Digital Twin in Industry 4.0: Advantages

Predictive Maintenance:

- Gaining a holistic view of the health and performance of equipment, companies can immediately detect anomalies and deviations in its operations.
- Maintenance and replenishment of spare parts can be proactively planned to minimize time-to-service and avoid costly asset failures.
- Predictive maintenance using Digital Twins can provide a new service-based revenue stream while helping improve product reliability.

Process Planning and Optimization:

- A digital footprint ingesting sensor and ERP data of a manufacturing line can comprehensively analyze performance.
- This helps diagnose the root cause of inefficiencies and throughput losses, optimizing yields and reducing wastes.
- Rich, integrated historical data on equipment, processes, and environments can enable downtime forecasting to improve production scheduling.

Product Design and Virtual Prototyping:

- Virtual models of in-use products provide comprehensive insights into usage patterns, degradation point, workload capacity, incurring defects.
- By better understanding a product's characteristics and failure modes, designers and developers can correctly evaluate product usability and improve future component design.
- Digital twin technology additionally aids in developing virtual prototypes and running robust simulations for feature testing based on empirical data.

ADVERSARY SIMULATION AS A METRIC

Metrics

- How many way there are to build an intrusion
- How long does it take to implement an intrusion
  - *Min*
  - *Average*
  - *Deviation*
  - *Number of attacks*
  - *Time*
- Black swan intrusions
- Worst case attacker
- TTPs and intrusions …

To compute these metrics we need information about attack paths, attack graphs …

Attack path

- A sequence of actions such that the success of the sequence enables the attacker to implement an intrusion ie to reach a goal from its attack surface
- The sequence can be restricted:
  - *neglect some actions according to the analysis and the detail level of interest*
  - *include only successful attacks or even failed attack execution*
- We can map the sequence into
  - *a sequence of system modules that are the targets of the attacks in the sequence*
  - *a sequence of security states, where each state is described as a set of access rights and pieces of information the attacker owns*
  - *a subsequence with all and only the actions useful to reach the goal*
- A useless attack grants rights the attacker does not use to reach a goal
- A plan is the subset of a sequence that includes useful attacks only
- There may be distinct paths and plans to the same goal

Paths and strategy

- Paths
  - *can be discovered by adversary emulation*
  - *the paths of an attacker depend upon its strategy*
- We can stop an attacker if we know not only its actions but also how it moves in a system = we need to know its strategy
- The information on the strategy and the adversary simulation is the only way to discover paths due the huge complexity of “static” solutions
- The discovery of all the attack paths may require
  - *to automate the execution of an intrusion*
  - *to target some digital twins of the target*

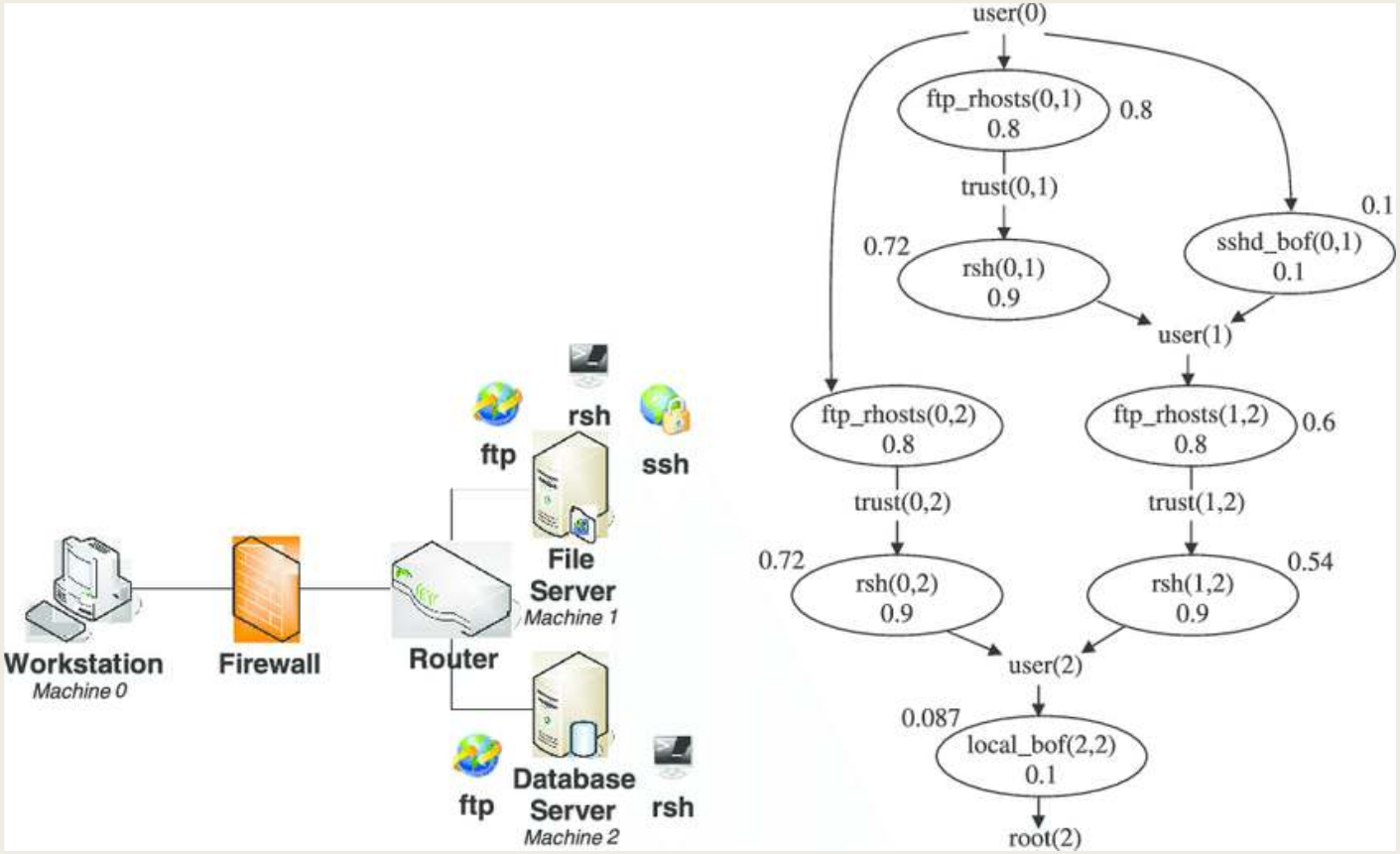
Paths and vulnerability ranking

- An intrusion requires a path not just one attack
- A path includes distinct attacks
- There may distinct paths to the intrusion goal
- A vulnerability per se does not result in a risk for a system
- The risk arises because of the joint presence of several vulnerabilities
- This implies that ranking the vulnerabilities of a system according to the risk EACH of them poses is meaningless.
- A coherent scoring is system (context) dependent
- Ranking each vulnerability is the goal of the Common Vulnerability Scoring System CVSS where Common = System independent

Attack graph

- It represents the alternative paths an attacker builds in its intrusions
- A huge number of alternative definitions but we can abstract them by saying that an attack graph is an oriented, labelled graph where
  - *Nodes represents security states*
  - *Links represents (are labelled with) the actions (the Ps in TTP) that enable the attacker to move from one state to the following one*
  - *The source node represents the initial rights and information of the attacker*
  - *Each final node (no output arcs) corresponds to the attacker goals*
    Distinct final nodes always exist if the attacker has distinct goals
- The graph
  - *is acyclic because no action can result in the loss of rights or of information*
  - *may include cycle of length 1 if we aims to model the failure of attacks*
  - *arcs may be labelled with the probability the action is successful*

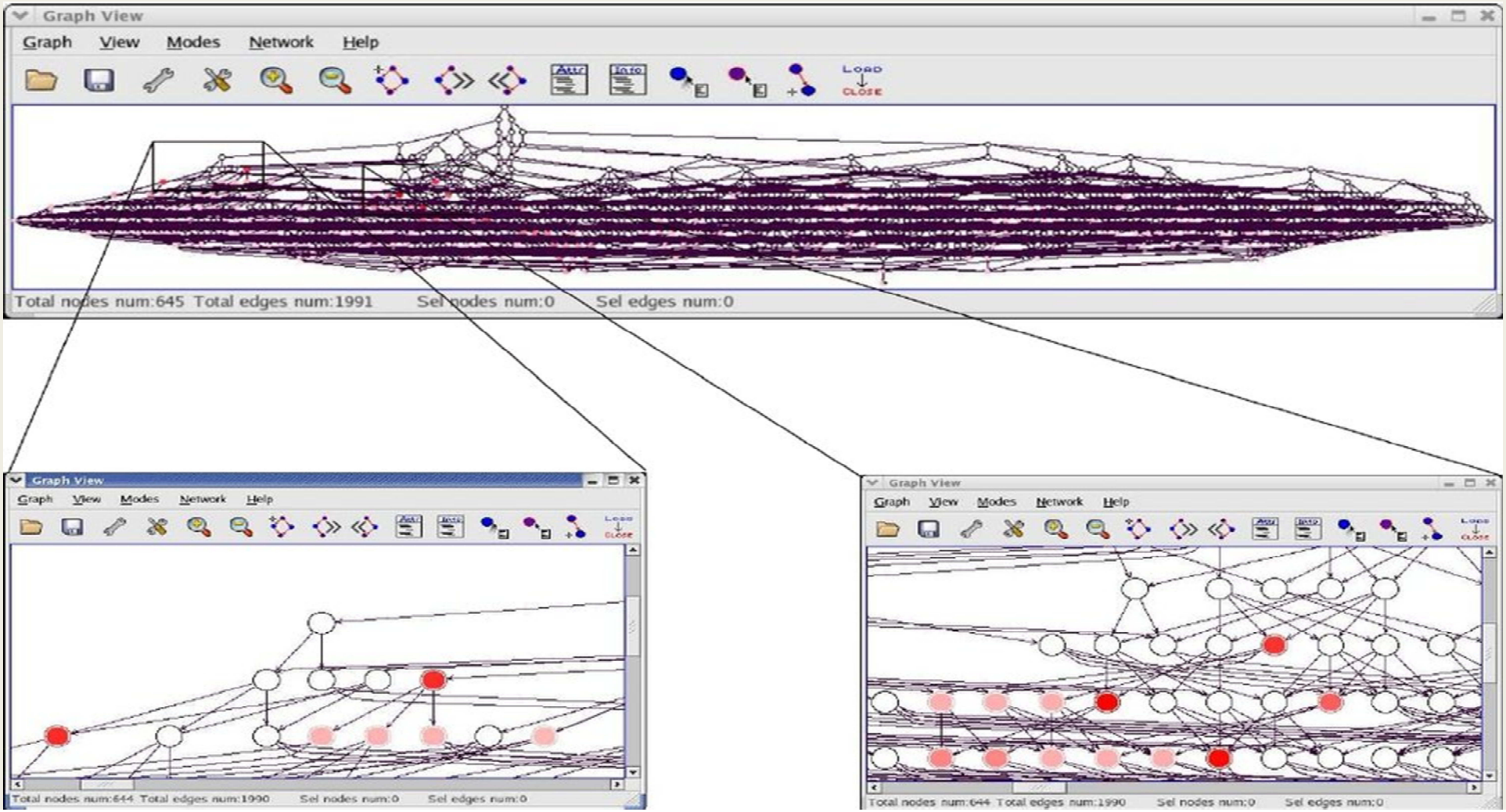
Attack graph



Attack graph

- A very nice formalism, but it cannot model any reasonable system due to the exponential increase in the number of states (nodes)
- Building an attack graph results in unmanageable due to the monotonic increase of access rights and to the large number of permutations. As an example, consider that
  - *an attack can be executed in a state (node) N then it can also be executed in any state (node) that can be reached from N*
  - *M attacks can be executed in N then the graph should represent any of the M! permutations of the M attacks*
- We cannot build a complete attack graph and then prune some of its paths to describe a strategy and the corresponding action sequences
- The only solution generates the alternative sequences by adversary emulation and merges them into a graph

Attack graph



Attack graph and strategies

- The only feasible approach to build an attack graph is to generate it
  - *emulate the strategy of an attacker*
  - *produce the possible sequences*
  - *merge the sequences into a graph*
- The generation considers
  - *the attacker priorities when selecting an action*
  - *the success/failure of an attack to handle it according to the attacker strategy*
- Multiple simulations can discover all the attacker sequences or at least all its plans
- Strategies can be ranked according to the number and/or the properties of the sequences they generate
- Interested to generate the largest number of successful sequences

Generated attack graph

- Using the generated attack graph we can compute
  - *the risk due to an attacker according to the number of sequences and the success probability of each one*
  - *the time to reach a goal according to the number of successes and of failures in a sequence*
  - *the probability of choosing a sequence by a Monte Carlo method ie by multiple simulation*
- Interesting problems
  - *How we can be sure that all the plans have been discovered?*
  - *Can we design a strategy that generates all the plans?*
  - *Can we use the graph to improve the system?*

Generate all the sequences

- Multiple simulations = a Monte Carlo method
- A strategy that random selects one of the possible attacks can discover all the sequences given enough time
- The time or the number of attempts the attacker has can reduce the number of simulation and the complexity of each simulation
- The number of sequences and the time to compute them is a first evaluation of the robustness of the target system
- Honeypot = a fake system with a huge number of vulnerabilities deployed to slow down and detect the intrusion
- Unwilling honeypot = a real system that is fully of vulnerability and slows down the attacker due to the time to select attacks

Improve the system

- Consider CGPA the complete attack graph generated using plans only, ie all the sequences are plans, and only the attacks (privilege escalation, lateral movement, …) in the plan
- We can compute CS a cut set of CGPA such that
  - *By removing arcs in CS we partition CGPA into some subgraphs*
  - *One subgraph includes the source node of CGPA*
  - *Further subgraphs include the destination nodes of CGPA*
- Each arc of CGPA = attack, we remove an arc by removing the vulnerability that enable the attack
- By removing arcs in Cs we prevent the attacker from reaching its goals
- A large number of defense strategies can be modelled as a percolation of the attack graph

Improve the system ?

- We are sure that the updated system has a better robustness only if we have discovered all the attack plans
- If we cut just some of the path, this may result in a system that is weaker
  - *Assume the attacker applies a really bad strategy that returns with a higher probability the plans with a lower success probability*
  - *Obviously, the attacker is unaware of this property otherwise, being intelligent, it would change it*
  - *If we stop the plans that the attacker follows with the largest probability, we force the attacker to follow the best attack plans*
- This paradox is know as Braess paradox because it has been discovered by Braess a logistic researcher while investigating why the building of a new bridge on the Thames reduced the average traffic speed

Braess Paradox

- **Braess's paradox** is the observation that adding one or more roads to a road network can slow down overall traffic flow through it.
- The paradox was postulated in 1968 by German mathematician Dietrich Braess, who noticed that adding a road to a particular congested road traffic network would increase overall journey time.
- The paradox may have analogies in electric grid and biological systems. It has been suggested that in theory, the improvement of a malfunctioning network could be accomplished by removing certain parts of it.
- The paradox has been used to explain instances of improved traffic flow when existing major roads are **closed**
- The last case is the interesting one for our problem: closing a path improve the situation for other paths

A first consequence …

- Several security standards and best practices requires the execution of a penetration test
- A penetration test is a simulated attack where the owner pays some people to find vulnerabilities in its system and then use them to attack the system
- Then these vulnerabilities are removed
- The assumption is that the attacker is ethic and will not hide some vulnerabilities it has discovered to exploit for personal profit
- Anyway, even if we believe the attacker is ethic ☺ due to time and financial constrains, the ethical attacker can find just a few attack paths
- Closing this path may reduce the system robustness and result in a worst security
- Big organization have a dedicated team of people (red team) to attack the system and remove the vulnerabilities they find

An even more important consequence…

- Most papers on security assume a non decreasing relation between the security investment and the resulting overall robustness
- As an example
  - *if you adopt C_1 countermeasure then the robustness changes from R to $R+D(C_1)$*
  - *If you adopt $C_1+ C_2$ countermeasures then the robustness changes from R to $R+D(C_1+ C_2)$*
  - *Everyone assumes that $D(C_1+ C_2) \geq D(C_1)$*
  - *but because of Braess paradox it may happen that $D(C_1+ C_2) < D(C_1)$*
- The only possible solution is *trust but verify* that is to measure the robustness of the system after deploying the countermeasures. This strongly increase the cost of the countermeasure due to the need to check their effectiveness

Discovering all the paths: automation

- The discovery requires a huge number of simulations to cover all the possible combinations of actions and action outputs, ie success or failure
- A first solution to automate the intrusion is the adoption of an attack platform
- An attack platform is a software tools to automate all/some steps of an intrusion
- Starting from a database of actions and of exploits, the platform can implement an intrusion in two ways
  - *The platform discovers and implements the best sequence of actions for the intrusion*
  - *The platform "import" the sequence of action to implement the intrusion*

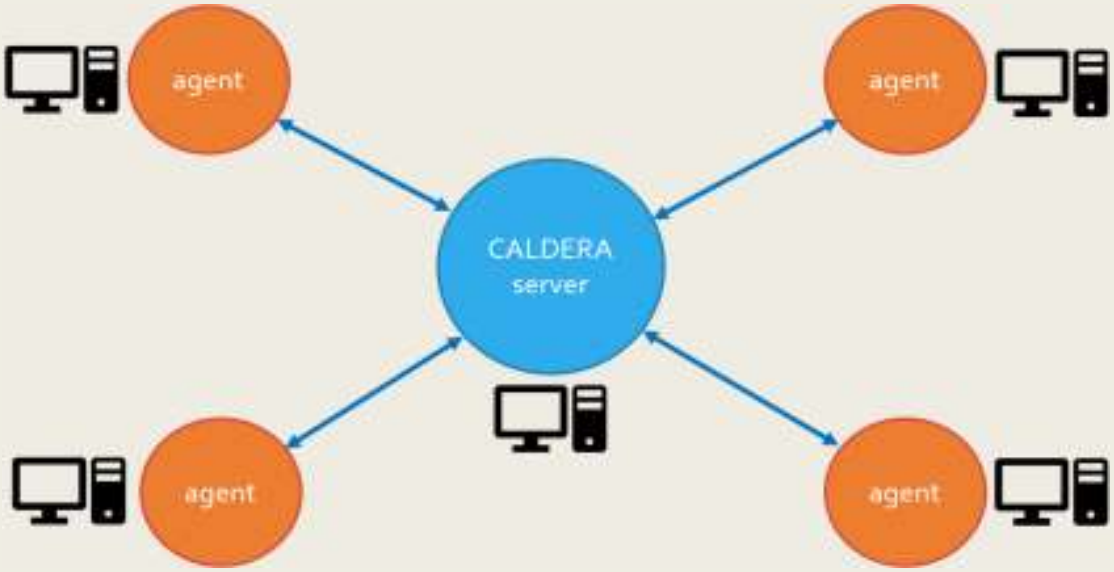
Caldera/MITRE

- CALDERA (Cyber Adversary Language and Decision Engine for Red Team Automation) is a platform to automate practices such as adversary emulation and red-teaming that is to attack YOUR system
- MITRE has created the platform to reduce the resources to run routine tests that evaluate the robustness of a system
- The platform supports the testing of security solutions adopted by the endpoints and the evaluation of the defensive posture of a network with respect to the TTPs of the ATT&CK Matrix
- Focusing on post-compromise scenarios, CALDERA neglects the actions prior to the initial violation and it assumes that the attacker is already present (has a foothold) within the network.
- CALDERA uses
  - the ATT&CK language for adversary representation (adversary profile),
  - a planner to process collected information and choose subsequent actions
  - an agent to execute operations

Caldera

- Implemented in Python
- The simulation of an adversary profile (= an intrusion) is an *operation*
- An operation can be structured into *phases*.
- Each phase involves the execution of an *ability* = a code fragment (a P in TTP)
- Abilities in distinct phases communicate through *facts*
  - *the first one produces a fact,*
  - *the second one uses information in a fact to initialize some variables in its code fragment*
- A centralized architecture, based on
  - *A server that administers the operations, plans the actions to be executed in the simulations and makes available to the user all the information from by the various agents*
  - *Client-side agents. Each simulates the actual presence of an attacker on the host that runs the agent itself*
- After planning the server waits for the agents to connect and distributes the corresponding action

Caldera Architecture

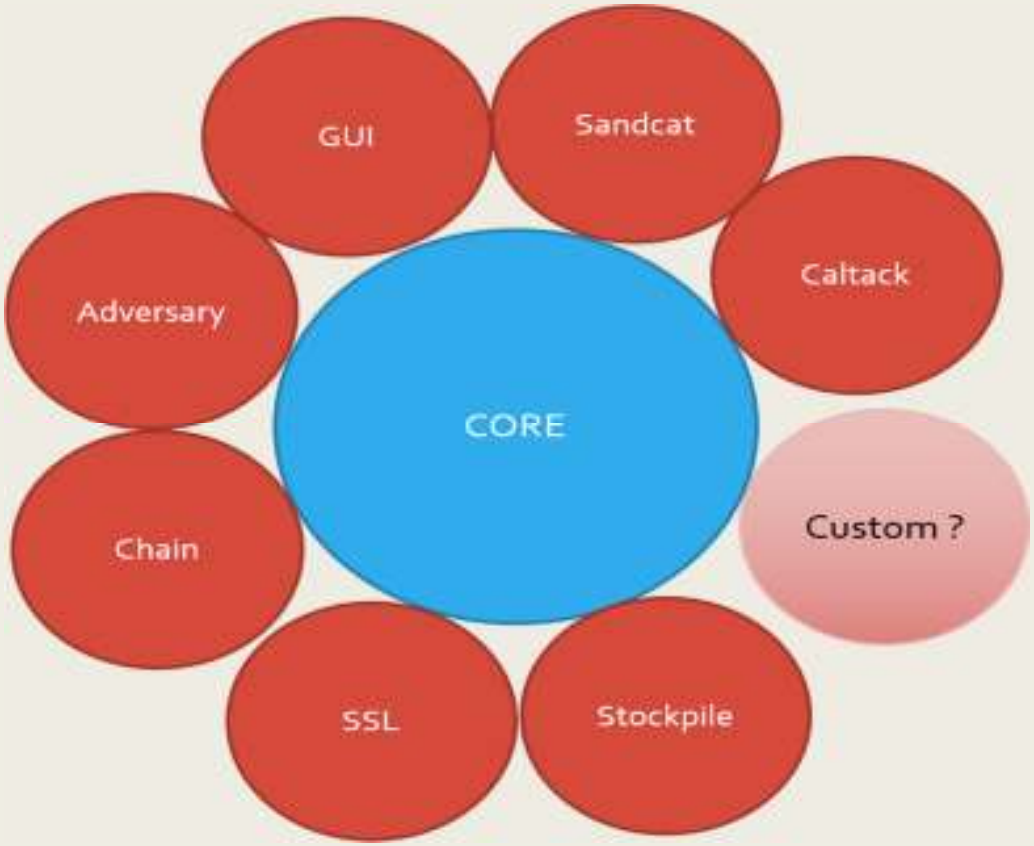


No command and control infrastructure, this shows that the final goal is not to attack a real system but Just to check if the intrusion is detected

Caldera – short history

- 2017 first version with an AI planning strategy but with significant limitations
  - *it could only operate in Windows networks and*
  - *complex installation of server and client side.*
  - *client installation required the CAgent agent and a remote RAT trojan.*
  - *The RAT executed the actions and it was started manually or following lateral movements.*
  - *at most 20 hosts involved in an intrusion because the planning was sophisticated but slow*
- 2019, second version that maintained only the centralized structure and references to the ATT&CK framework but with high customization to simplify both installation and use. Two main changes
  - *a plug-in architecture, with a core of functionality and basic services, such as database management and authentication.*
  - *A chain mode mode that enables users to orchestrate and chain atomic units into attack sequences. The new planning is less dynamic and more dependent on the user ability to create adversary profiles*
- Third version some days ago, with the ability to import adversary emulation plan that is intrusion programmed from outside the platform

Caldera Plug In Architecture



Caldera – Main features

- An *agent* is a software module that does not require an installation.
  - *It connects to the server at regular intervals (by default every 60 seconds) and it sends a keepalive signal.*
  - *If the agent belongs to at least one active operation, the server responds with the instructions to be executed.*
  - *After the execution, the agent returns the server any result.*
  - *An agent that never connects to the server will never receive the actions to execute.*
- The Sandcat plug-in (54ndc47) is the default agent. Other agents may be implemented that communicate with the server through the protocol defined by CALDERA.
- A *group* is a set of agents to execute operations on multiple nodes in parallel. Upon startup, an agent joins the group specified in the startup command. The user can change an agent group at any time.
- An *ability* is an implementation of an ATT&CK technique. Since the same technique can use distinct tools and procedures, distinct abilities can implement the same technique. Abilities are stored in YML format within the Stockpile plug-in, together with the adversary profiles that use them.

Caldera: an ability

```yaml
- id: 9a30740d-3aa8-4c23-8efa-d51215e8a5b9
  name: Scan WIFI networks
  description: View all potential WIFI networks on host
  tactic: discovery
  technique:
    attack_id: T1016
    name: System Network Configuration Discovery
  platforms:
    darwin:
      sh:
        command: |
          ./wifi.sh scan
        payload: wifi.sh
    linux:
      sh:
        command: |
          ./wifi.sh scan
        payload: wifi.sh
    windows:
      psh:
        command: |
          .\wifi.ps1 -Scan
        payload: wifi.ps1
```

Mitre Att&ck Matrix

Implementations in distinct platforms

Caldera: an ability

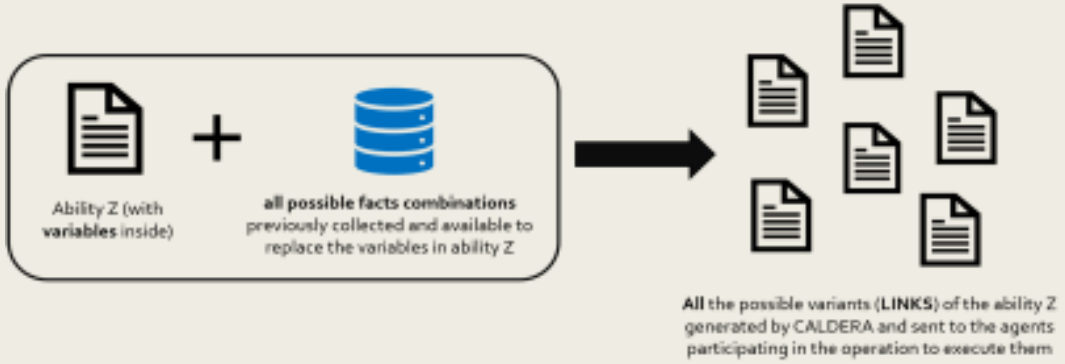
- Each ability is associated with a set of platforms that describes the operating systems it can affect.
- For each platform, some performers exist eg sh for Darwin and Linux powershell for Windows
- Four sections for each platform a mandatory one, *Command*, and three optional, *Payload*, *Cleanup* and *Parser*.
- Command stores the user to run and that may contain some variables. Three default global variables:
  - *#{server}: the server network address. It determines the correct server location for each agent.*
  - *#{group}: the agent group. Useful in lateral movement where the Command section tries to start a new agent and insert it in the ongoing operation.*
  - *#{location}: the agent location on the client file system.*
- The server replaces global variables with their value before sending the *Command* section to the agent. Alternatively, CALDERA can use facts = information provided by the user or returned a previous ability.
- The Payload section lists the files to execute the ability. In the example the Windows payload is "wifi.ps1 that the agent can download "wifi.ps1" from the server if not already present on the host.
- The Cleanup section restores the system state to the one before executing Command. i.e. if Command creates a file, Cleanup can delete it.
- The Parser section analyzes the output of the command section and transform it into one or more facts.

Caldera: a fact

- A *fact* is an information about a node and is related to the variables the abilities in the Command section. Three of its elements are:
  - *Property: a three-part descriptor that identifies the type of fact. For example, "host.user.name" indicates that the fact value is a username. This format supports the specification of the main ("host"), minor ("user") and specific ("name") fact components*
  - *Value: an arbitrary string. An appropriate value for "host.user.name" can be "Administrator" or "John".*
  - *Score: an integer, default 1, denoting the fact relative importance. If a fact score is less than or equal to 0, the fact is blacklisted and cannot be used.*

Caldera: a bit of confusion

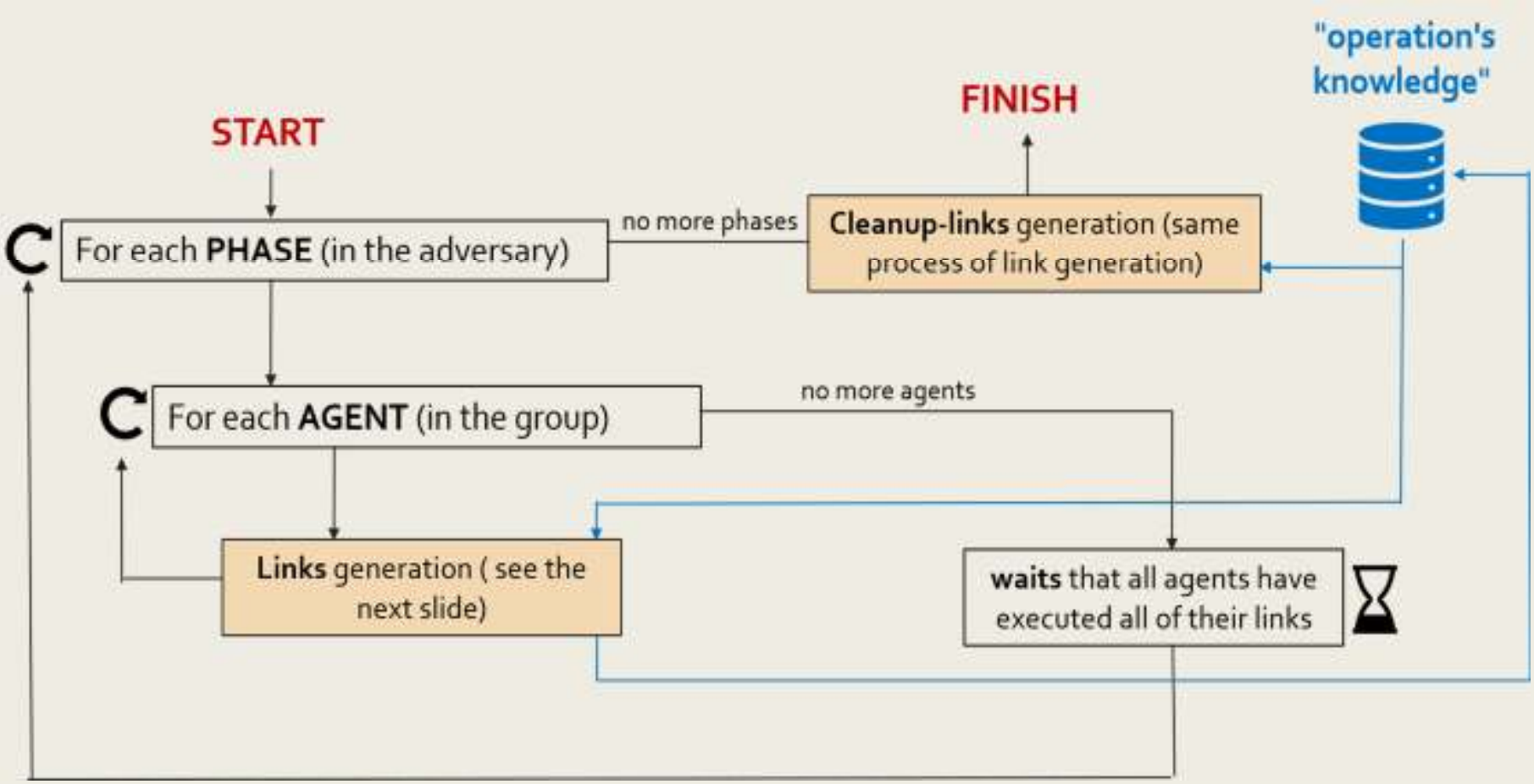
- When the server sends an ability to the agents, it scans the Cleanup commands and instructions to identify any variables and checks whether it can replace anyone using the facts it has. Variables that match facts based on ownership are replaced.
- The server creates one instance of each cleanup command/instruction for each combination of the facts it has that matches with the variables.
- Each distinct combination is associated with a score that is the cumulative score of all the facts used to complete the command.
- The variants thus generated are performed in decreasing score order. This procedure is executed for each agent in an operation. This can result into ambiguous and counterintuitive scenarios.



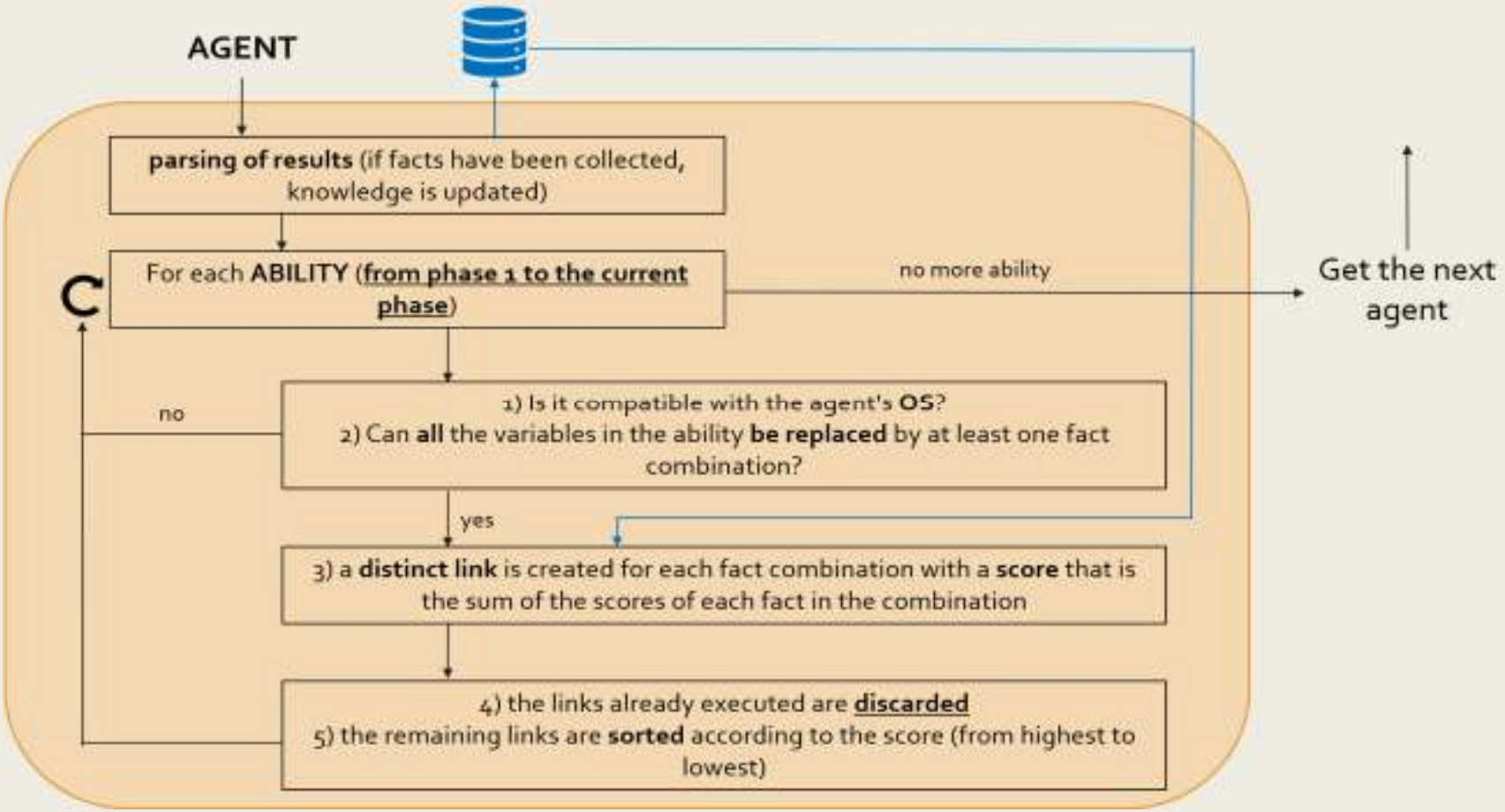
Caldera adversaries and plans

- An *adversary profile* (or adversary) represents an attacker as a set of abilities grouped into phases.
- A *planner* is the decision-making process that selects the abilities to use and in what order to implement a single phase.
- The planner creation is simple because it only implements the *execute*() function that CALDERA invokes to execute the individual phases of an operation.
- A planner can discard an ability if does not cover the operating system of the agent node or if any combination of the fact does not match with all the ability variables.
- The *Sequential planner is the default planner*
  - *sends to each operation agent all the links generated from the ability in the executed phase,*
  - *sorts the links in a descending order according to the cumulative score of the facts they use*
  - *sends them to the agents and waits till all have completed the execution,*
  - *enables the user to choose the ability execution order when it builds the adversary profiles. This may result in the parallel execution of the operation on even a large number of agents.*

Caldera intrusion/operation



Caldera intrusion/operation



Caldera problems

1. The next step of an operation can begin only when all the agents have terminated the execution of the links they receives. The lack of a reply results in an infinite wait
2. No distinction between facts at the operation level (for all its agents) with respect to a local ones (for the single agent that collects an information). All the facts are global, and this causes the generation and execution of meaningless links for most agents.
3. The priority to the phases and then to the agents in an operation can cause ambiguities when a new agent joins an operation due to a successful lateral movement.
    1. *Even if the agent does not start from phase 1, it has to execute the links of all the abilities for all the operation phases from phase 1 to the current one.*
    2. *the order of the phases is not guaranteed, and some abilities may not be executed due to the lack of facts while other may be repeated or recovered in subsequent phases due to the creation of new facts.*

Caldera solutions

1. The later CALDERA versions have solved problems 1) and 2) by improving the characteristics of the facts and by introducing the concept of *trust* for agents.
2. In these versions, a fact is no longer global, and it can be local to a host. This makes it possible to generate the correct links only.
3. An agent that does not connect to the server for a preconfigured time, it will be labeled as *untrusted*, and the operation will continue without waiting for its replies.
4. *rules* have been introduced to discard some of the facts produced.
5. New plug-ins that implement a reverse shell or can improve the robustness of a node to attacks by installing agents that perform "defensive" profiles. Another plug-in implements the initial attack to penetrate the target system.

(2+3 output of a thesis in this department)

Caldera 3.0

- **Emu** a plugin that imports adversary emulation plans from a library of adversary emulation plans to allow organizations to evaluate their defensive capabilities against the real-world threats they face.
- Emulation plans are an essential component in testing current defenses for organizations to prioritize their defenses around actual adversary behavior. A set of common emulation plans that are available to all enables organizations to use their limited time and resources to focus on understanding how their defenses actually fare against real-world threats.

Emulation Plan	Intelligence Summary
FIN6	FIN6 is thought to be a financially motivated cyber-crime group. The group has aggressively targeted and compromised high-volume POS systems in the hospitality and retail sectors since at least 2015...
APT29	APT29 is thought to be an organized and well-resourced cyber threat actor whose collection objectives appear to align with the interests of the Russian Federation...
menuPass	menuPass is thought to be threat group motivated by collection objectives, with targeting that is consistent with Chinese strategic objectives...

Caldera 3.0

Peer-to-Peer Communication

- It allows agents within internal networks to chain together to enable beaconing and communications where a direct connection is not possible. The implementation in sandcat allows for varied channels of communication as well, so that an agent can be configured for the environment is being deployed
- Function to discover of peers, so that an agent can be deployed from a generic binary and discover if there are any available peers to connect out through if direct connection to the C2 server is not possible.

Lateral Movement Tracking

- Adds in the capability for caldera to track lateral movement. A link is passed in as an optional command line argument when executing an agent.

Uploads

- file uploads and exfiltration were performed using hardcoded commands (curl, powershell webclient, etc) that required HTTP(s) connection to the C2. If the agent is using peer-to-peer and cannot directly access the server, old file upload commands wouldn't work as intended.
- By adding in the upload capability as a separate ability, agents use their contact method's built-in upload functionality to send file upstream, whether it is directly to the C2 server or to another agent proxy peer

Deadman Abilities

- Users can now specify deadman abilities so that agents run them prior to termination. Use case for this functionality is to specify an ability to remove the agent executable once the agent terminates, or other defense evasion abilities like clearing log

Metasploit Framework: an offensive platform

What is the Metasploit Framework?

- *According to the Metasploit Team;*

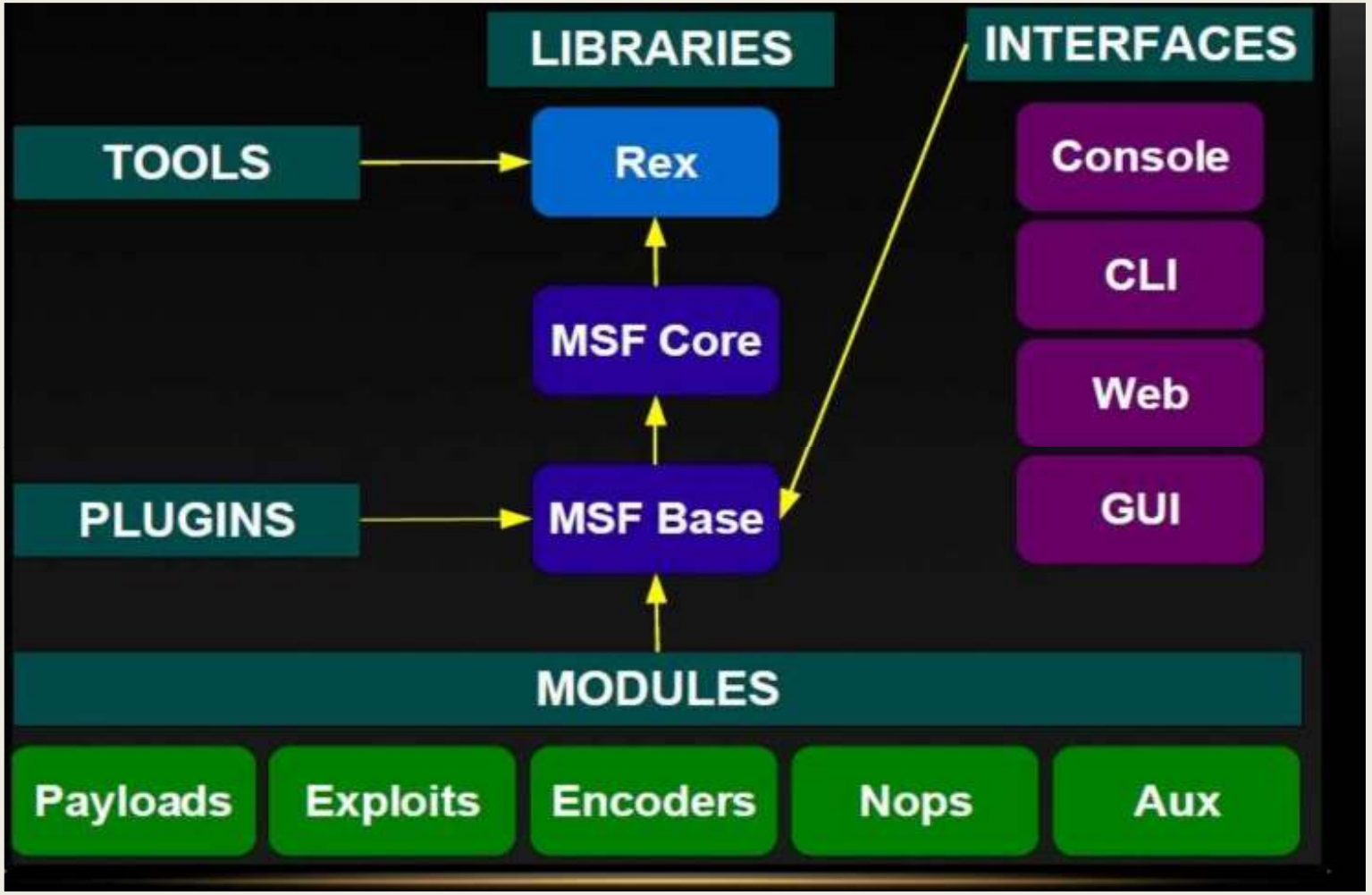
> *"The Metasploit Framework is a platform for writing, testing, and using exploit code. The primary users of the Framework are professionals performing penetration testing, shellcode development, and vulnerability research."*

Understanding MSF

- The MSF is not only an environment for exploit development but also a platform for launching exploits on real-world applications. It is packaged with real exploits that can provide real damage if not used professionally.

- The fact that MSF is an open-source tool and provides such a simplified method for launching dangerous attacks, it has and still is attracting wannabe hackers and script kiddies that do no more than create additional problems on networks and system.

MSF architecture



Some components

- Exploit
  - Active : will exploit a specific host, run until completion, and then exit
  - Passive = listener: waits for incoming hosts and exploits them as they connect.
    - It almost always focuses on clients such as web browsers, FTP clients, etc.
    - In can be used in conjunction with email exploits, waiting for connections.
- Payload
  - Singles are payloads that are self-contained and completely standalone. A Single payload can be something as simple as adding a user to the target system or running calc.exe.
  - Stagers setup a network connection between the attacker and victim and are designed to be small and reliable.
- Encoder it generates a payload for a language by removing some characters
- Nops it adds a nop sled to a payload (useful in buffer overflow to overcome missing information on the memory map

Meterpreter (The Meta-Interpreter)

- Advanced, dynamically extensible payload that uses in memory dll injection and extended over the network at runtime through pivoting
- An extendable platform for creating a post attack way to implement complex features that cannot be run in assembly.
- It is not just limited to the built-in command functions. You can create your own modules to use on the target system.
- It operates inside the target process but upon examination a system it will not be discovered
- It works in a client ↔ server configuration. Where the server mearly acts as a communication and loading mechanism . A protocol is designed to handle this communication

Attack platform

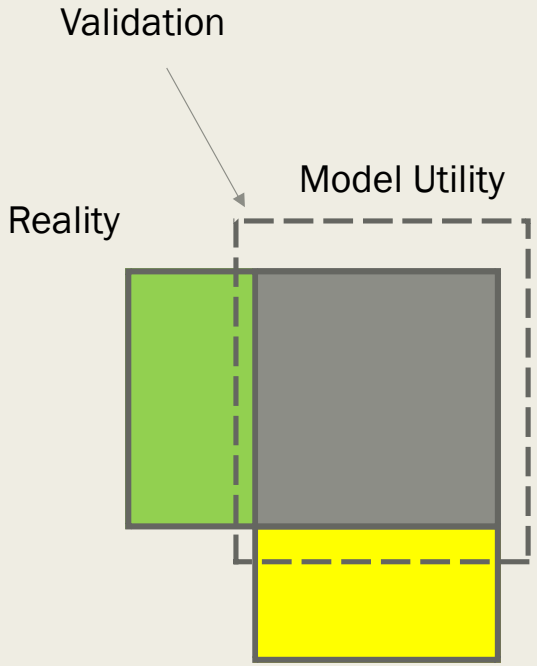
- The main problem is the strategy that is how to compose the actions/abilities into an intrusion
- The sequence could be discovered or precomputed by another platform after collecting the information about the target system

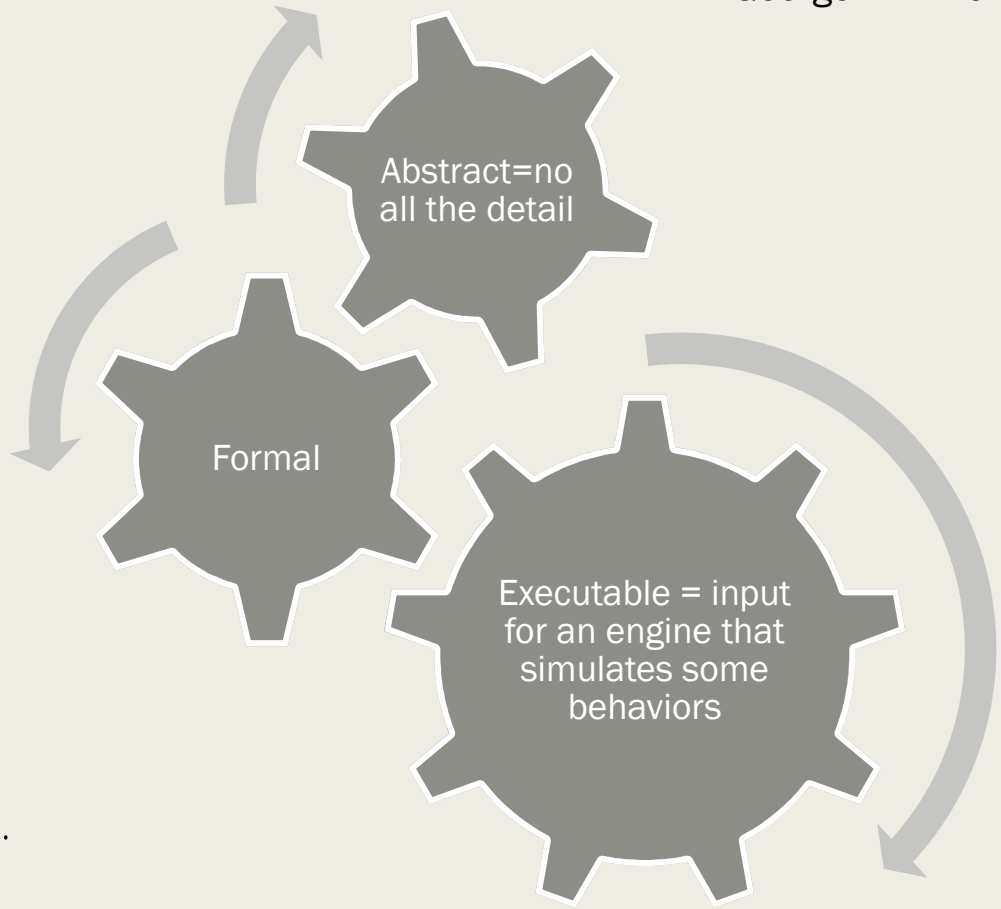
Discovering all the paths: noise and twins

- Digital twin = model of the target system with the information for adversary emulation
- A hybrid model merging experimental and formal properties
- Information of a digital twin
  - *Modules*
    - Physical modules
    - System modules
      - *OSes*
      - *VMs*
      - *Hypervisors*
    - Applications
  - *Physical connections*
  - *Logical connections*
  - *Filtering and routing rules*
  - *Vulnerabilities & Attack: attack attributes to discover attack sequences*
  - *Dependencies = information flows*

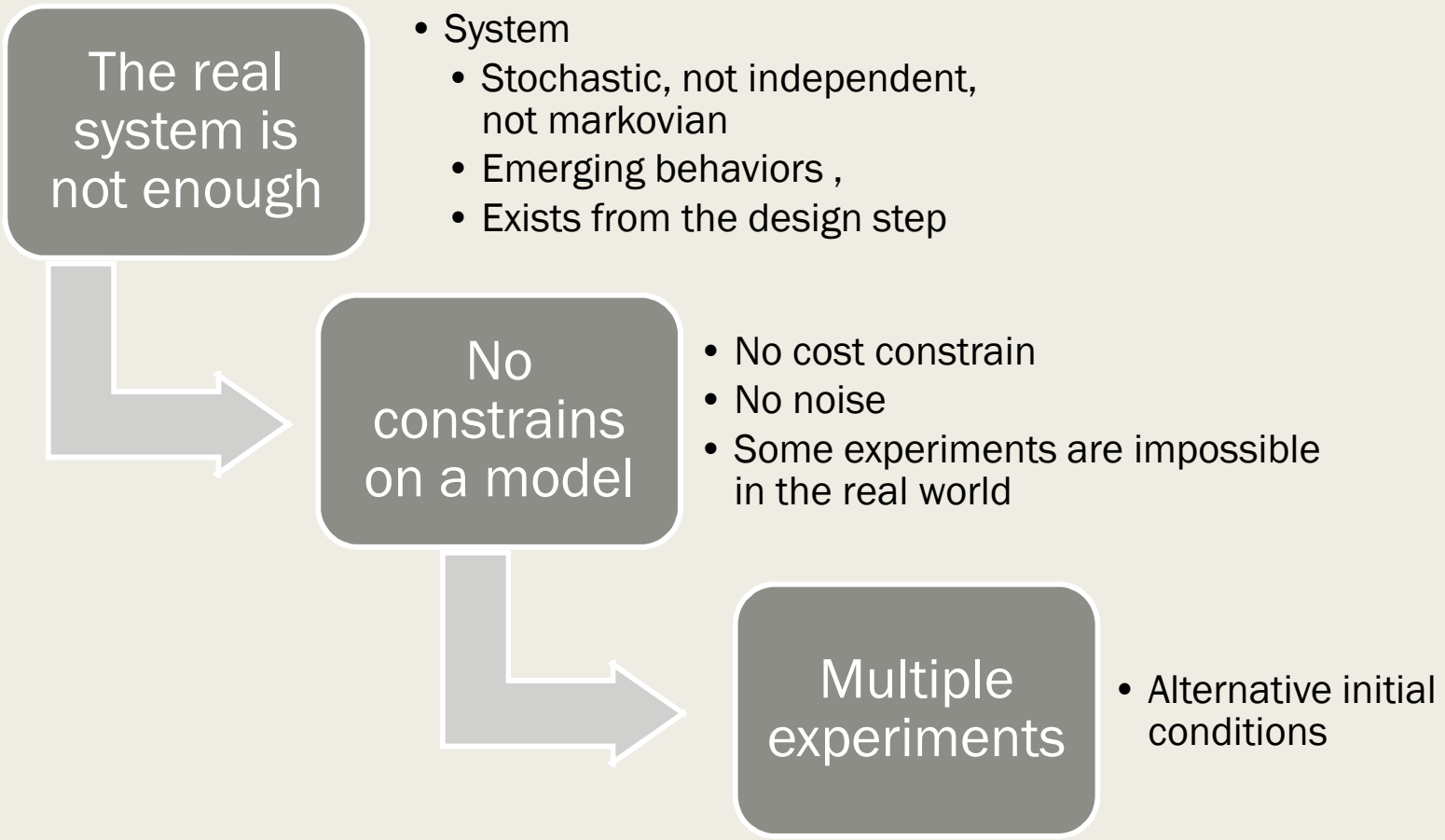
All models are wrong but some are useful …

George E. P. Box





Model Advantages



The twin and the plan

- The emulation on the twin can be used to compute the optimal plan
- This leaves open the problem of how to explore the target system to discover paths, optimal paths etc.
- An alternative is learning to attack on a set of cases and then apply the learned strategy to any other system (transfer learning)
- The learned strategy could be improved after the intrusion