CALDERA

Automating Adversary Emulation

Andy Applebaum, Doug Miller

The MITRE Corporation

MITRE

Approved for Public Release. Distribution Unlimited. Case Number 17-4315

Bios

Andy Applebaum

Lead Cyber Security Engineer

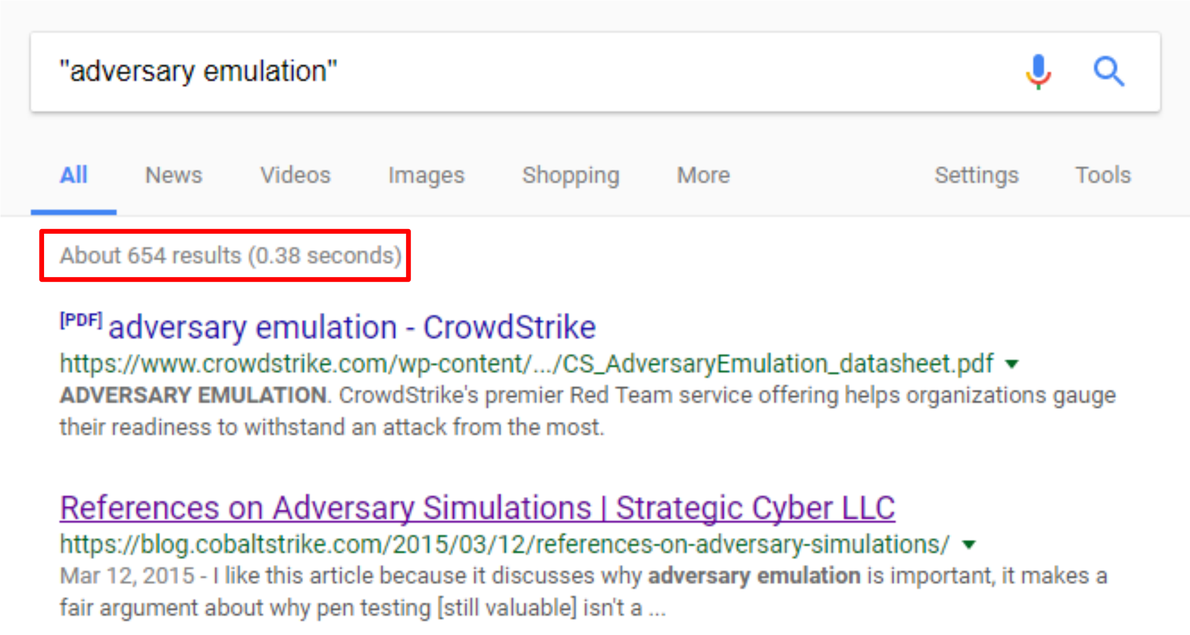
ATT&CK, AI Research

Doug Miller

Sr. Cyber Security Engineer

ATT&CK, CARET, Red teaming

Adversary Emulation?



About 654 results (0.38 seconds)

[PDF] adversary emulation - CrowdStrike
https://www.crowdstrike.com/wp-content/.../CS_AdversaryEmulation_datasheet.pdf
ADVERSARY EMULATION. CrowdStrike's premier Red Team service offering helps organizations gauge their readiness to withstand an attack from the most.

References on Adversary Simulations | Strategic Cyber LLC
https://blog.cobaltstrike.com/2015/03/12/references-on-adversary-simulations/
Mar 12, 2015 - I like this article because it discusses why **adversary emulation** is important, it makes a fair argument about why pen testing [still valuable] isn't a ...

The False Negative Problem

As a defender, you have no idea what you miss

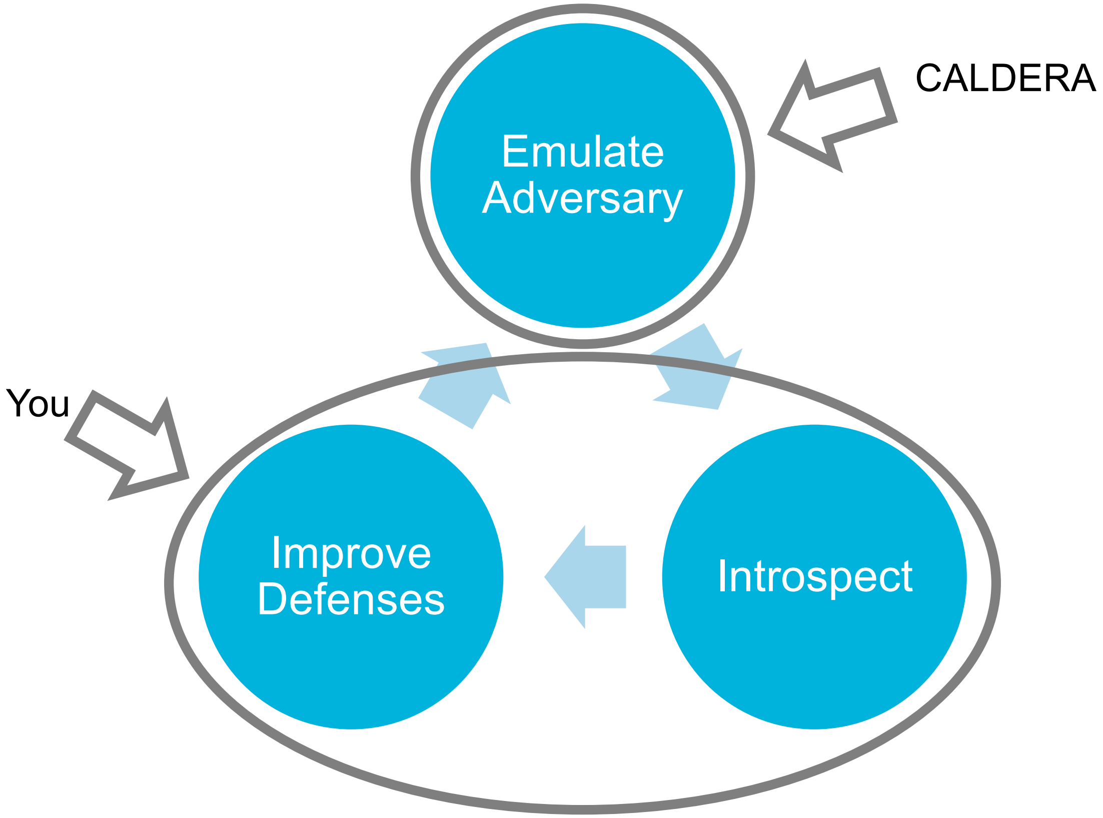
Cue: Adversary

- **Introduce a realistic* adversary on your network**

  *an emulated adversary

- ***Now you can determine what happens if an attacker gets on your network***
  - Did I detect them?
  - How far did they get?
  - How can I improve my detection and prevention?

Iterative Defensive Cycle



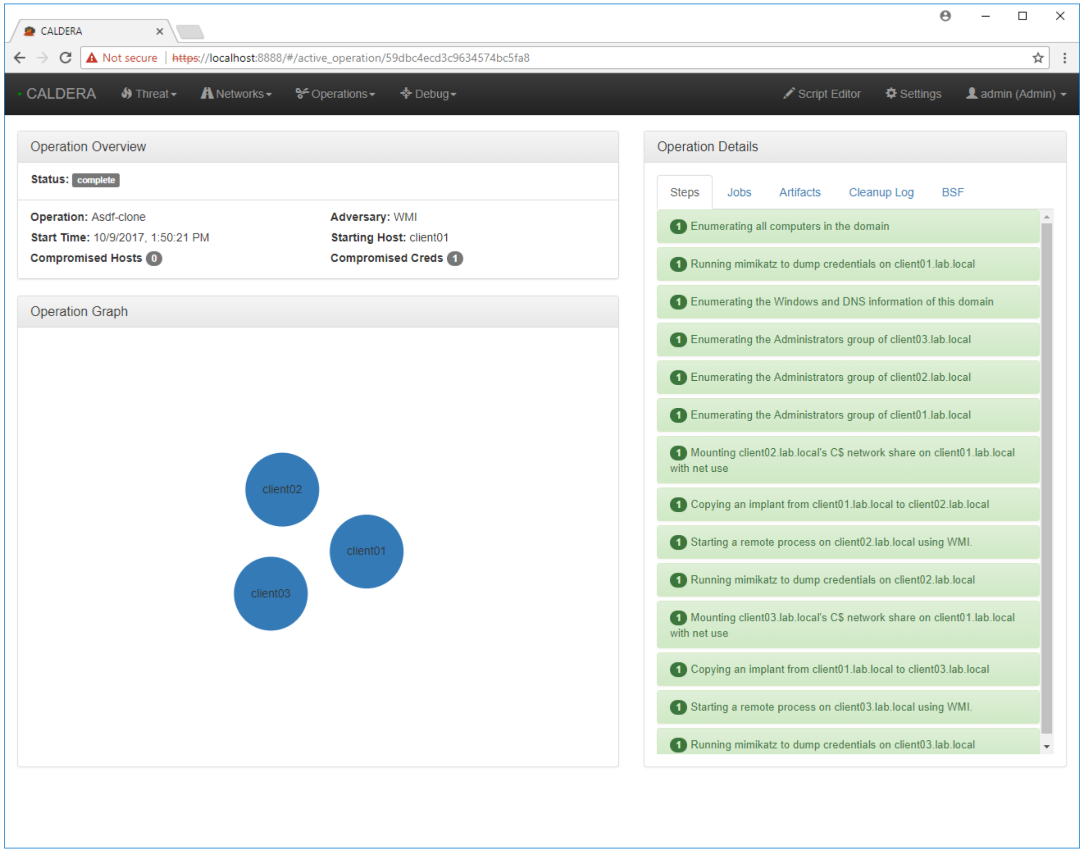
Successful Adversary Emulation

Make it real: Use the same techniques, tools, methods and goals of an attacker

End-to-End: Don’t just look for holes or perform small attacks. Start from the initial compromise and go until objectives are accomplished

Repeatable: Be repeatable, so that your detection and prevention improvement (or degradation) can be measured over time

CALDERA:



Operation Overview

Status: complete

Operation: Asdf-clone | **Adversary:** WMI
Start Time: 10/9/2017, 1:50:21 PM | **Starting Host:** client01
Compromised Hosts 0 | **Compromised Creds** 1

Operation Graph

Operation Details

1 Enumerating all computers in the domain

1 Running mimikatz to dump credentials on client01.lab.local

1 Enumerating the Windows and DNS information of this domain

1 Enumerating the Administrators group of client03.lab.local

1 Enumerating the Administrators group of client02.lab.local

1 Enumerating the Administrators group of client01.lab.local

1 Mounting client02.lab.local's C$ network share on client01.lab.local with net use

1 Copying an implant from client01.lab.local to client02.lab.local

1 Starting a remote process on client02.lab.local using WMI.

1 Running mimikatz to dump credentials on client02.lab.local

1 Mounting client03.lab.local's C$ network share on client01.lab.local with net use

1 Copying an implant from client01.lab.local to client03.lab.local

1 Starting a remote process on client03.lab.local using WMI.

1 Running mimikatz to dump credentials on client03.lab.local

CALDERA – Conducting an Operation

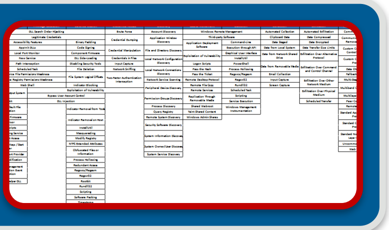
1. **Load the CALDERA shim onto network hosts**
2. **Create an adversary by giving it capabilities**
3. **Launch the operation**

- **During the operation:**
  - CALDERA will report its activities, including artifacts created
  - CALDERA will automatically stop if it has exhausted its toolkit
- **After the operation:**
  - CALDERA will provide a report of what it did
  - CALDERA will automatically "reset" infected hosts, removing artifacts and stopping processes

Ingredients for Automated Adversary Emulation

What the adversary can do

- The adversary model
- MITRE ATT&CK™



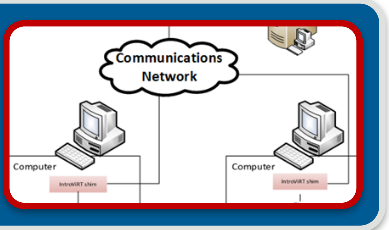
	DLL Search Order Hijacking		Brute Force	Account Discovery	Windows Remote Management		Automated Collection	Automated Exfiltration	Comm
	Legitimate Credentials		Credential Dumping	Application Window Discovery	Third-party Software		Clipboard Data	Data Compressed	Communi Remova
Accessibility Features		Binary Padding	Credential Manipulation	File and Directory Discovery	Application Deployment Software	Command-Line	Data Staged	Data Encrypted	Custom C Control
AppInit DLLs		Code Signing	Credentials in Files	Local Network Configuration Discovery	Exploitation of Vulnerability	Execution through API	Data from Local System	Data Transfer Size Limits	Custom C Pr
Local Port Monitor		Component Firmware	Input Capture	Local Network Connections Discovery	Logon Scripts	Graphical User Interface	Data from Network Shared Drive	Exfiltration Over Alternative Protocol	Data Ob
New Service		DLL Side-Loading	Network Sniffing	Network Service Scanning	Pass the Hash	InstallUtil	Data from Removable Media	Exfiltration Over Command and Control Channel	Fallback
Path Interception		Disabling Security Tools	Two-Factor Authentication Interception	Peripheral Device Discovery	Pass the Ticket	PowerShell	Email Collection	Exfiltration Over Other Network Medium	Multi-Stag
Scheduled Task		File Deletion		Permission Groups Discovery	Remote Desktop Protocol	Process Hollowing	Input Capture	Exfiltration Over Physical Medium	Multiband C
vice File Permissions Weakness		File System Logical Offsets		Process Discovery	Remote File Copy	Regsvcs/Regasm	Screen Capture	Scheduled Transfer	Multilaye
e Registry Permissions Weakness		Indicator Blocking		Query Registry	Remote Services	Regsvr32			Peer C
Web Shell	Exploitation of Vulnerability			Remote System Discovery	Replication Through Removable Media	Rundll32			Remote
tput System	Bypass User Account Control			Security Software Discovery	Shared Webroot	Scheduled Task			Standard Ap Pr
kit	DLL Injection			System Information Discovery	Taint Shared Content	Scripting			Standard C Pro
fault File tion		Indicator Removal from Tools		System Owner/User Discovery	Windows Admin Shares	Service Execution			Standard No Layer
Firmware		Indicator Removal on Host		System Service Discovery		Windows Management Instrumentation			Uncomm
isor		InstallUtil							Web
ripts		Masquerading							
ng Service		Modify Registry							
t Access		NTFS Extended Attributes							
Keys / Start er		Obfuscated Files or Information							
ort Provider		Process Hollowing							
dification		Redundant Access							
nagement tion Event ction		Regsvcs/Regasm							
		Regsvr32							
elper DLL		Rootkit							
		Rundll32							
		Scripting							
		Software Packing							
		Timestomp							

How the adversary chooses what to do

- CALDERA logic and decision engine

What the adversary needs to do it

- Infrastructure to support real adversary emulation
- Management server; client agents; web interface



The Adversary Model

Choosing an Adversary Model

CALDERA emulates a real adversary after they get into a network

ATT&CK Matrix™ Tactics and Techniques

<table>
<tr><th>Persistence</th><th>Privilege Escalation</th><th>Defense Evasion</th><th>Credential Access</th><th>Discovery</th><th>Lateral Movement</th><th>Execution</th><th>Collection</th><th>Exfiltration</th><th>Command and Control</th></tr>
<tr><td colspan="3">DLL Search Order Hijacking</td><td>Brute Force</td><td>Account Discovery</td><td colspan="2">Windows Remote Management</td><td>Audio Capture</td><td>Automated Exfiltration</td><td>Commonly Used Port</td></tr>
<tr><td colspan="3">Legitimate Credentials</td><td rowspan="2">Credential Dumping</td><td rowspan="2">Application Window Discovery</td><td colspan="2">Third-party Software</td><td>Automated Collection</td><td>Data Compressed</td><td rowspan="2">Communication Through Removable Media</td></tr>
<tr><td colspan="2">Accessibility Features</td><td>Binary Padding</td><td rowspan="2">Application Deployment Software</td><td>Command-Line</td><td>Clipboard Data</td><td>Data Encrypted</td></tr>
<tr><td colspan="2">AppInit DLLs</td><td>Code Signing</td><td rowspan="2">Credential Manipulation</td><td rowspan="2">File and Directory Discovery</td><td>Execution through API</td><td>Data Staged</td><td>Data Transfer Size Limits</td><td>Connection Proxy</td></tr>
<tr><td colspan="2">Local Port Monitor</td><td>Component Firmware</td><td rowspan="2">Exploitation of Vulnerability</td><td rowspan="2">Execution through Module Load</td><td>Data from Local System</td><td rowspan="2">Exfiltration Over Alternative Protocol</td><td rowspan="2">Custom Command and Control Protocol</td></tr>
<tr><td colspan="2">New Service</td><td>DLL Side-Loading</td><td>Credentials in Files</td><td rowspan="2">Local Network Configuration Discovery</td><td rowspan="2">Data from Network Shared Drive</td></tr>
<tr><td colspan="2">Path Interception</td><td>Disabling Security Tools</td><td>Input Capture</td><td>Logon Scripts</td><td>Graphical User Interface</td><td rowspan="3">Exfiltration Over Command and Control Channel</td><td rowspan="2">Custom Cryptographic Protocol</td></tr>
<tr><td colspan="2">Scheduled Task</td><td>File Deletion</td><td>Network Sniffing</td><td rowspan="2">Local Network Connections Discovery</td><td>Pass the Hash</td><td>InstallUtil</td><td rowspan="2">Data from Removable Media</td></tr>
<tr><td colspan="2">File System Permissions Weakness</td><td rowspan="2">File System Logical Offsets</td><td rowspan="3">Two-Factor Authentication Interception</td><td>Pass the Ticket</td><td>MSBuild</td><td>Data Encoding</td></tr>
<tr><td colspan="2">Service Registry Permissions Weakness</td><td>Network Service Scanning</td><td>Remote Desktop Protocol</td><td>PowerShell</td><td>Email Collection</td><td rowspan="2">Exfiltration Over Other Network Medium</td><td>Data Obfuscation</td></tr>
<tr><td colspan="2">Web Shell</td><td>Indicator Blocking</td><td rowspan="2">Peripheral Device Discovery</td><td>Remote File Copy</td><td>Process Hollowing</td><td>Input Capture</td><td>Fallback Channels</td></tr>
<tr><td rowspan="2">Authentication Package</td><td colspan="3">Exploitation of Vulnerability</td><td>Remote Services</td><td>Regsvcs/Regasm</td><td>Screen Capture</td><td rowspan="2">Exfiltration Over Physical Medium</td><td>Multi-Stage Channels</td></tr>
<tr><td colspan="2">Bypass User Account Control</td><td></td><td rowspan="2">Permission Groups Discovery</td><td rowspan="2">Replication Through Removable Media</td><td>Regsvr32</td><td>Video Capture</td><td rowspan="2">Multiband Communication</td></tr>
<tr><td>Bootkit</td><td colspan="2">DLL Injection</td><td></td><td>Rundll32</td><td></td><td>Scheduled Transfer</td></tr>
<tr><td>Component Object Model Hijacking</td><td></td><td>Component Object Model Hijacking</td><td></td><td>Process Discovery</td><td>Shared Webroot</td><td>Scheduled Task</td><td></td><td></td><td>Multilayer Encryption</td></tr>
<tr><td>Basic Input/Output System</td><td></td><td>Indicator Removal from Tools</td><td></td><td>Query Registry</td><td>Taint Shared Content</td><td>Scripting</td><td></td><td></td><td>Remote File Copy</td></tr>
<tr><td>Change Default File Association</td><td></td><td>Indicator Removal on Host</td><td></td><td>Remote System Discovery</td><td>Windows Admin Shares</td><td>Service Execution</td><td></td><td></td><td>Standard Application Layer Protocol</td></tr>
<tr><td>Component Firmware</td><td></td><td>Install Root Certificate</td><td></td><td>Security Software Discovery</td><td></td><td>Windows Management Instrumentation</td><td></td><td></td><td>Standard Cryptographic Protocol</td></tr>
<tr><td>External Remote Services</td><td></td><td>InstallUtil</td><td></td><td>System Information Discovery</td><td></td><td></td><td></td><td></td><td>Standard Non-Application Layer Protocol</td></tr>
<tr><td>Hypervisor</td><td></td><td>Masquerading</td><td></td><td>System Owner/User Discovery</td><td></td><td></td><td></td><td></td><td>Uncommonly Used Port</td></tr>
<tr><td>Logon Scripts</td><td></td><td>Modify Registry</td><td></td><td>System Service Discovery</td><td></td><td></td><td></td><td></td><td>Web Service</td></tr>
<tr><td>Modify Existing Service</td><td></td><td>MSBuild</td><td></td><td>System Time Discovery</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Netsh Helper DLL</td><td></td><td>Network Share Removal</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Redundant Access</td><td></td><td>NTFS Extended Attributes</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Registry Run Keys / Start Folder</td><td></td><td>Obfuscated Files or Information</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Security Support Provider</td><td></td><td>Process Hollowing</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Shortcut Modification</td><td></td><td>Redundant Access</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Windows Management Instrumentation Event Subscription</td><td></td><td>Regsvcs/Regasm</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Winlogon Helper DLL</td><td></td><td>Regsvr32</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>Rootkit</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>Rundll32</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>Scripting</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>Software Packing</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>Timestomp</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

https://attack.mitre.org

Supported Adversary Actions

Persistence

- Registry autorun keys
- Scheduled Task
- Services

Privilege Escalation

- Weak service perms
- Weak service file perms
- Unquoted paths (Path interception)

Defense Evasion

- Scripting
- Timestomping

Credential Access

- Credential Dumping

Lateral Movement

- Remote File Copy
- Windows Admin shares
- Pass the Hash
- PsExec

Discovery

- Remote System Discovery
- Local Network config
- Registry
- Account
- System information
- Processes/services
- System Owner
- Permission Group
- Files

Execution

- PowerShell
- Scheduled Task
- WMI
- SC (service controller)

Exfiltration

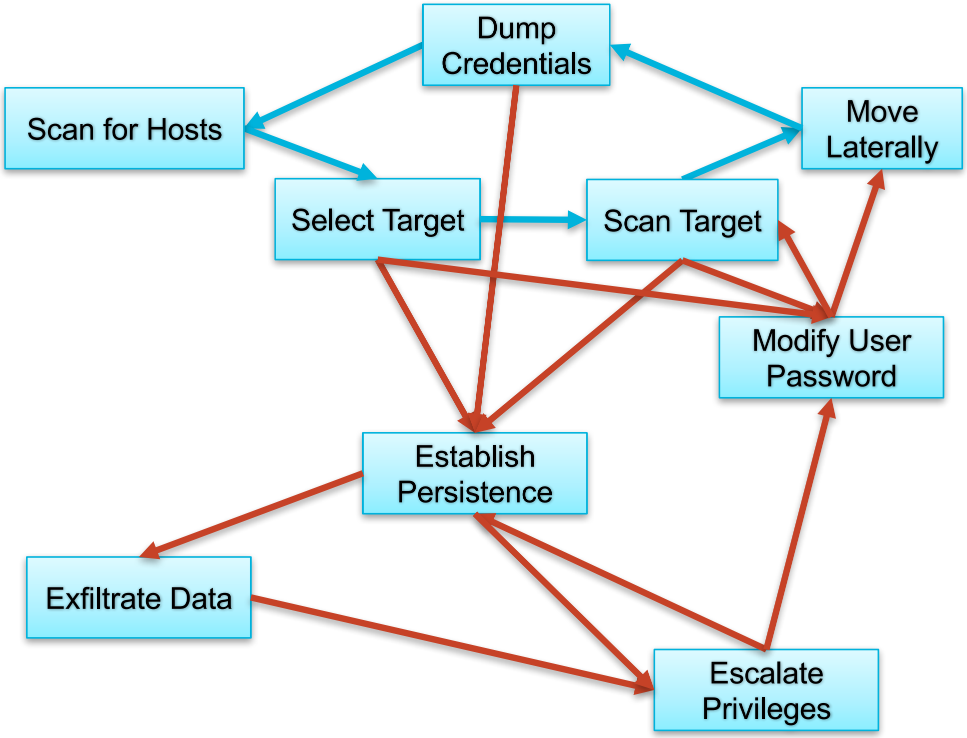
- HTTP/S

Decision Making for Automated Adversary Emulation

Early CALDERA

- **First version**
  - Finite-state machine (FSM) approach
  - Successfully tested within MITRE

- ***Hard to extend to new techniques***
  - Action needs to be coded into FSM
  - FSM logic needs to be recomputed
  - Inflexible in operation; hard to configure

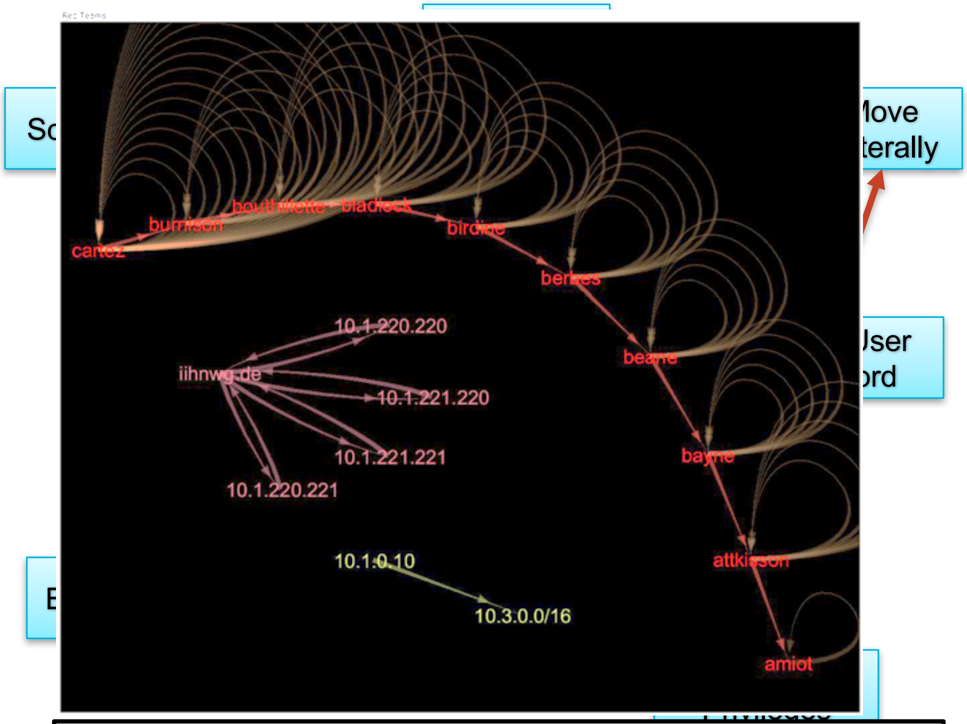


Early CALDERA

- **First version**
  - Finite-state machine (FSM) approach
  - Successfully tested within MITRE

- ***Hard to extend to new techniques***
  - Action needs to be coded into FSM
  - FSM logic needs to be recomputed
  - Inflexible in operation; hard to configure

- ***Predictability during execution***
  - Easy to spot and identify



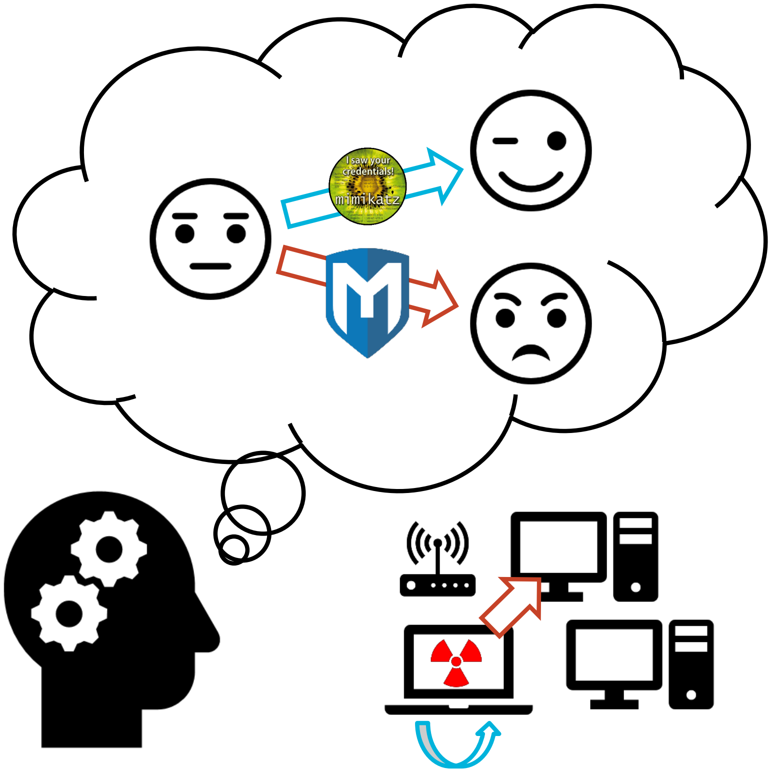
Visualized with the MITRE CyGraph tool

Noel, S., et al. "CyGraph: Graph-Based Analytics and Visualization for Cybersecurity." *Handbook of Statistics* 35 (2016): 117-167.

Designing an Adversary Decision Engine

- **Typical engagements have *human* operators dictating and controlling the assessment**
  - Targeting, TTP selection, domain inference…
  - … all needs to be fully automated!

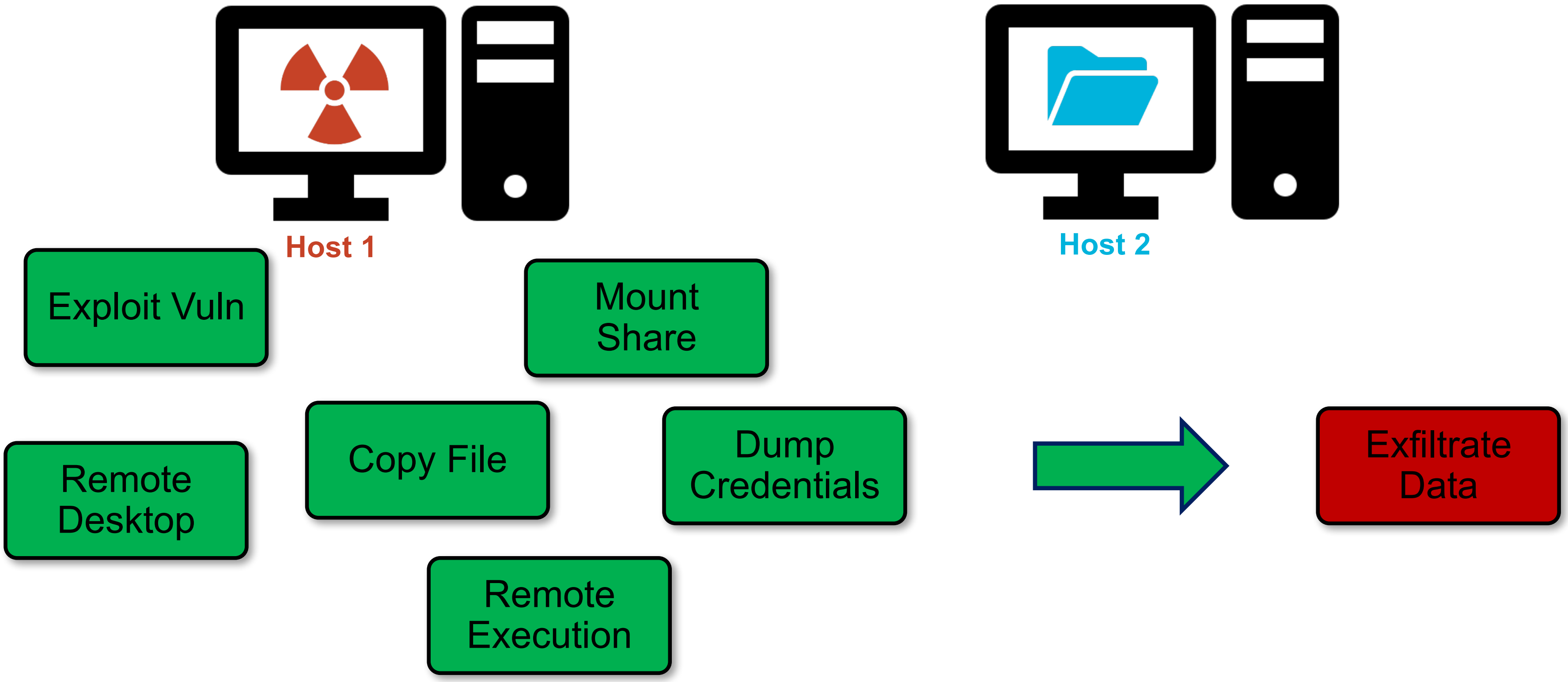
- **Ideally, our automated adversary will:**
  - Make intelligent choices
  - Achieve tactical objectives
  - Easily incorporate new techniques
  - Work in new and unknown environments
  - Vary operations to test the defense
  - Chain weaknesses for maximum effect



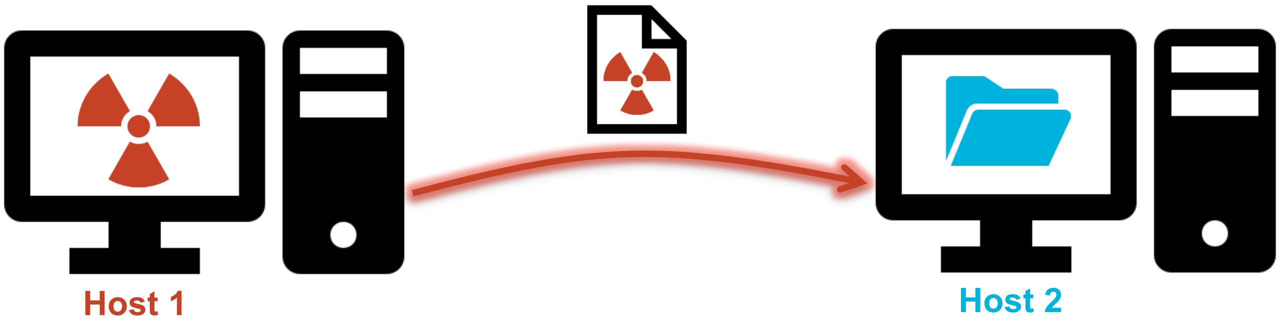
Example Scenario

- Start with code execution and a RAT on Host 1
- Identified sensitive files on Host 2
- *Goal*: exfiltrate sensitive data from Host 2

Example Scenario



Analyzing Copying Over a File



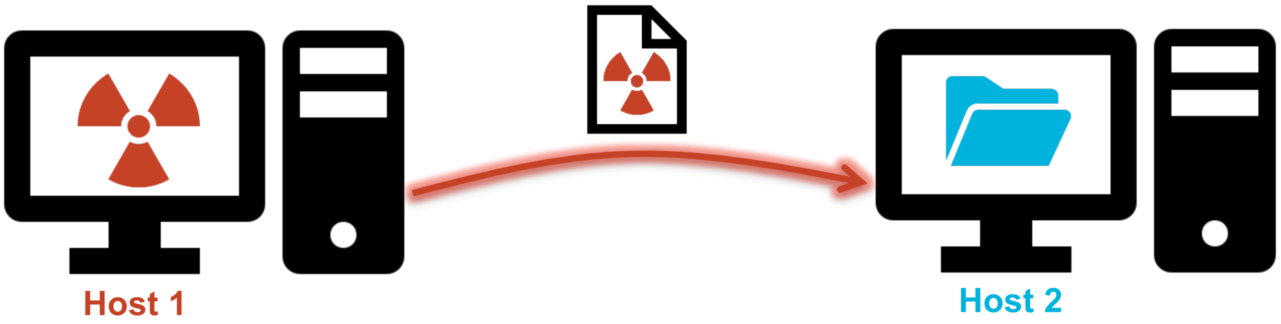
- **What do we need to do to copy a RAT over?**
  - Working RAT on source host
  - Mounted file share from target onto source host
  - Write access to file share

Requirements, or *preconditions*

- **What happens after copying a RAT over?**
  - There will be a new file on the target host
  - That file will contain the RAT

Consequences, or *postconditions*

Making a Plan to Copy a File



Dump Credentials

- **Need**: Elevated RAT
- **Get**: Credentials

Mount Share

- **Need**: Credentials
- **Get**: Mounted Share

Copy File

- **Need**: Mounted Share
- **Get**: File on Target

Sequence of actions, or *plan*

The Core CALDERA Idea

- **Move from an *explicit, prescribed* decision model towards a *dynamic, model-based one***

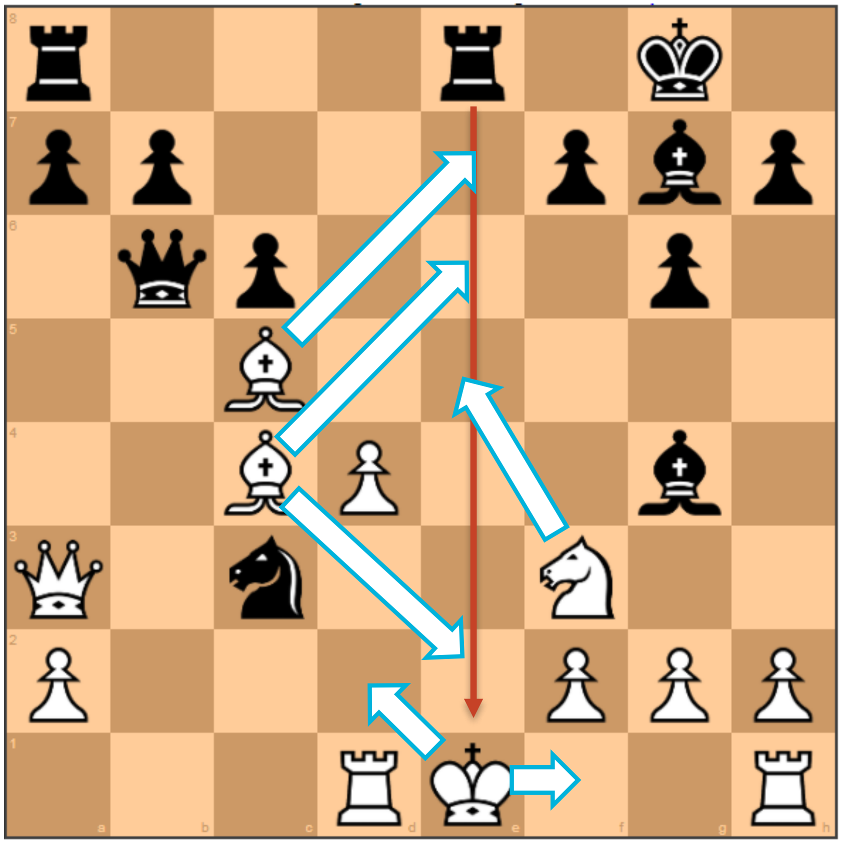
- **Tag actions with *semantic execution information*:**
  - **Preconditions** specify the requirements that must be true to execute a technique
  - **Postconditions** specify the consequences that will be true after executing a technique

- **No longer need to be explicitly told what to do!**
  - Instead, compare the current state to the available actions to determine which are valid

- **Added bonus: planning for the future**
  - If I dump credentials now, that can help me execute lateral movement in the future!

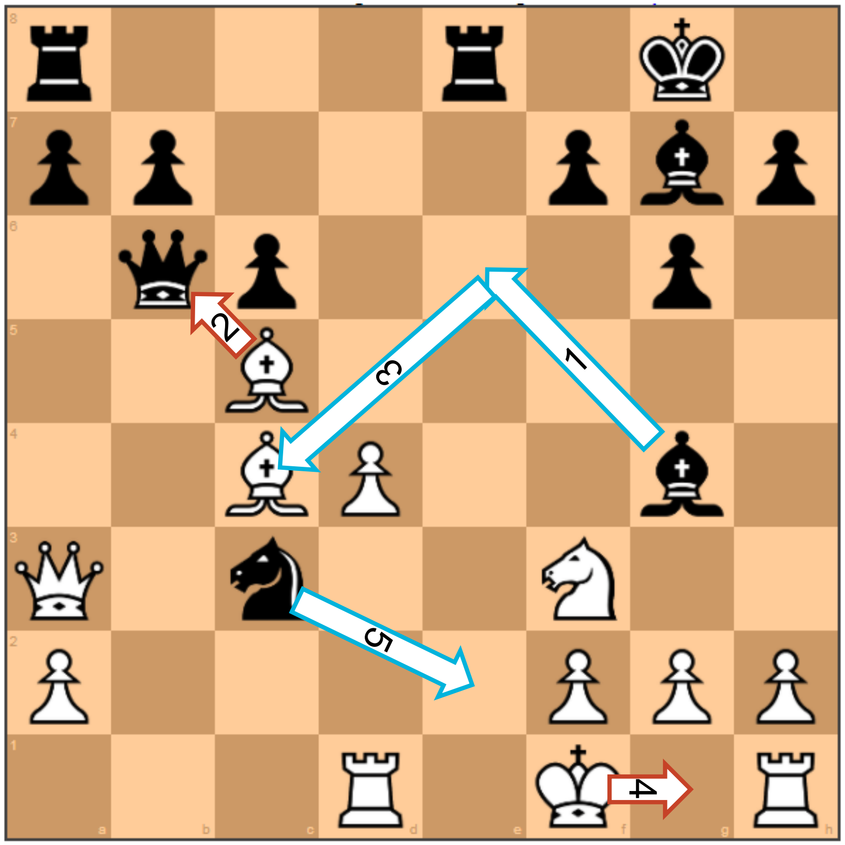
Fun With Preconditions

- **Preconditions tell you what you can do *now***
  - In chess: can tell you which moves are valid
  - Taken further: can tell you which moves are legal
- **In the emulation sense: given an escalated foothold on a host, we can:**
  - Dump credentials
  - Add/modify registry keys
  - Setup scheduled tasks
  - …



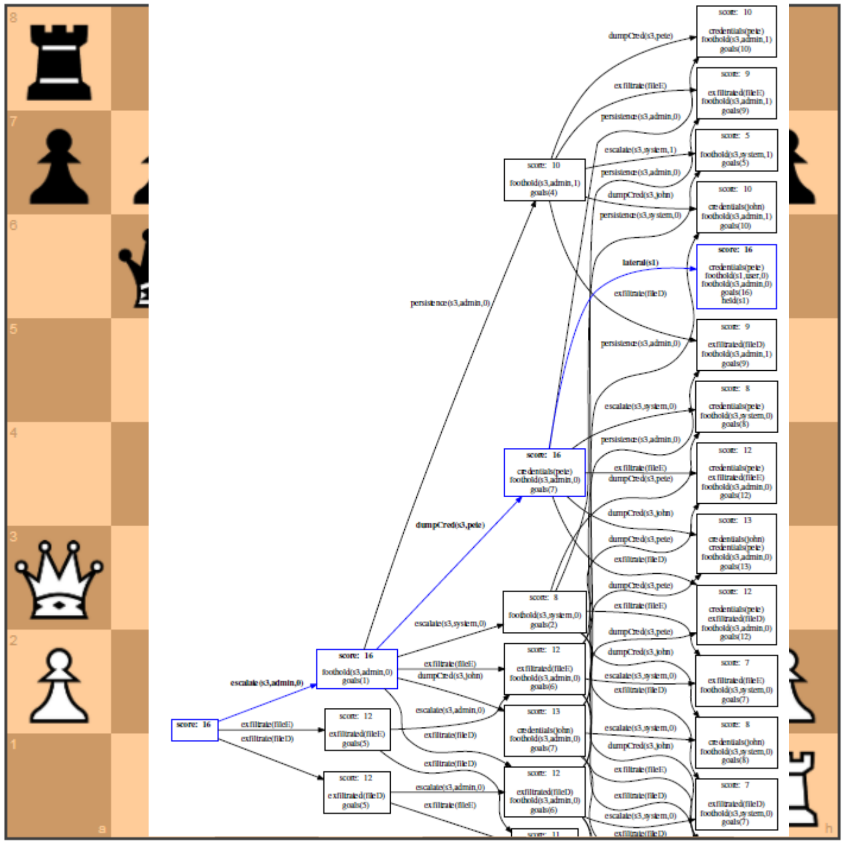
Fun With Postconditions

- **Postconditions tell you what will be true *after***
  - With preconditions, can *chain actions together* to plan for the future
  - Can evaluate *potential futures* to select the best immediate action

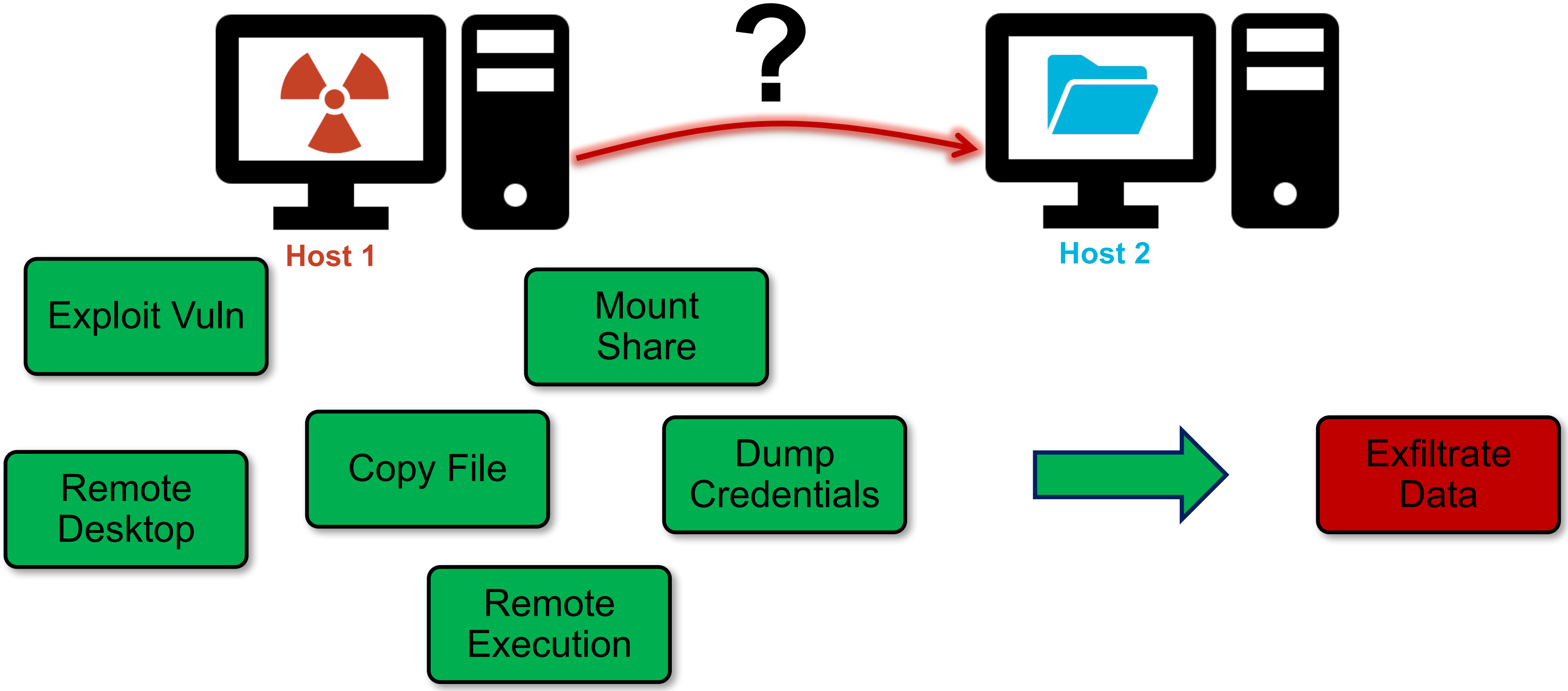


Fun With Postconditions

- **Postconditions tell you what will be true *after***
  - With preconditions, can *chain actions together* to plan for the future
  - Can evaluate *potential futures* to select the best immediate action
- **In the emulation sense: given an escalated foothold on a host, we can:**
  - Dump credentials and then laterally move
  - Add/modify registry keys and then dump credentials
  - Setup scheduled tasks and then add/modify registry keys
  - …



Making Progress



?

Host 1

Host 2

Exploit Vuln

Mount Share

Remote Desktop

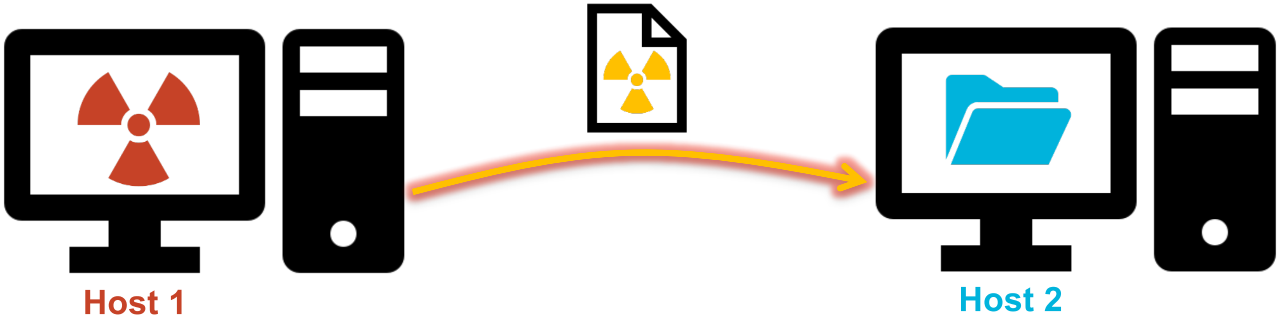
Copy File

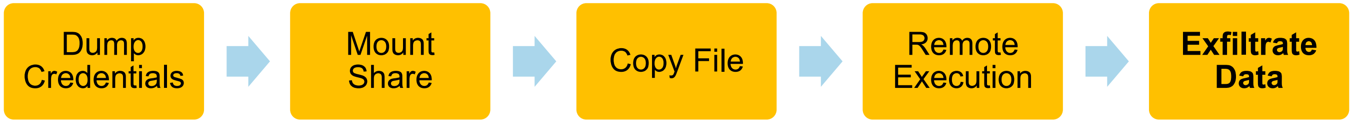
Dump Credentials

Exfiltrate Data

Remote Execution

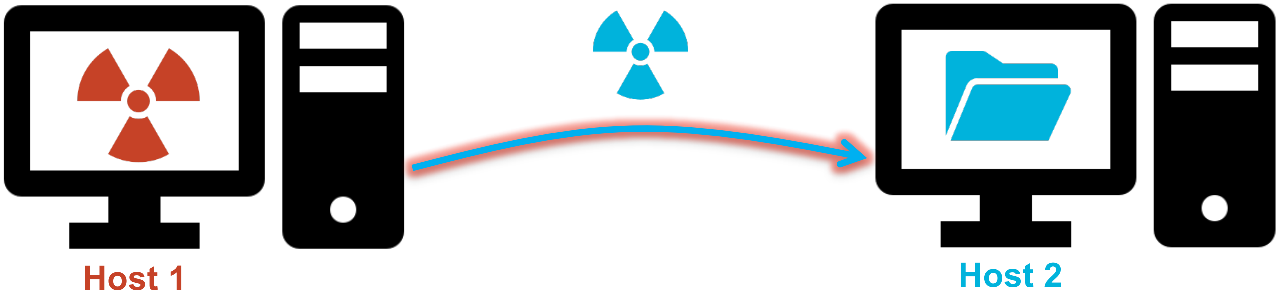
Building Plans: Copying a File





Dump Credentials → Mount Share → Copy File → Remote Execution → **Exfiltrate Data**

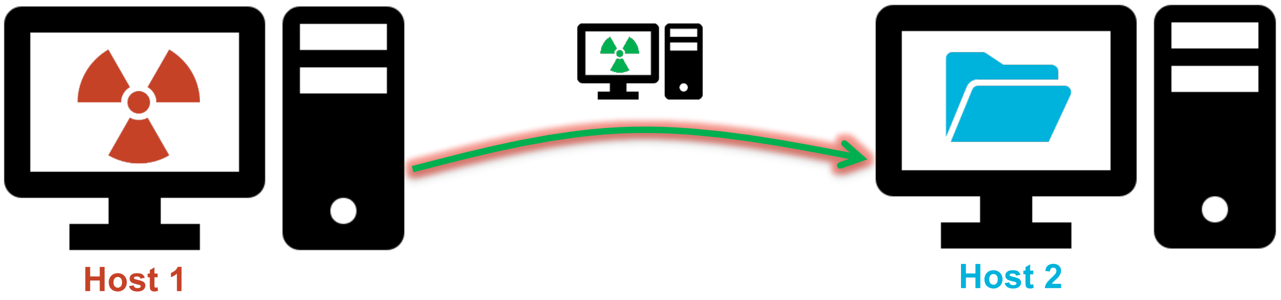
Building Plans: Exploiting a Vulnerability

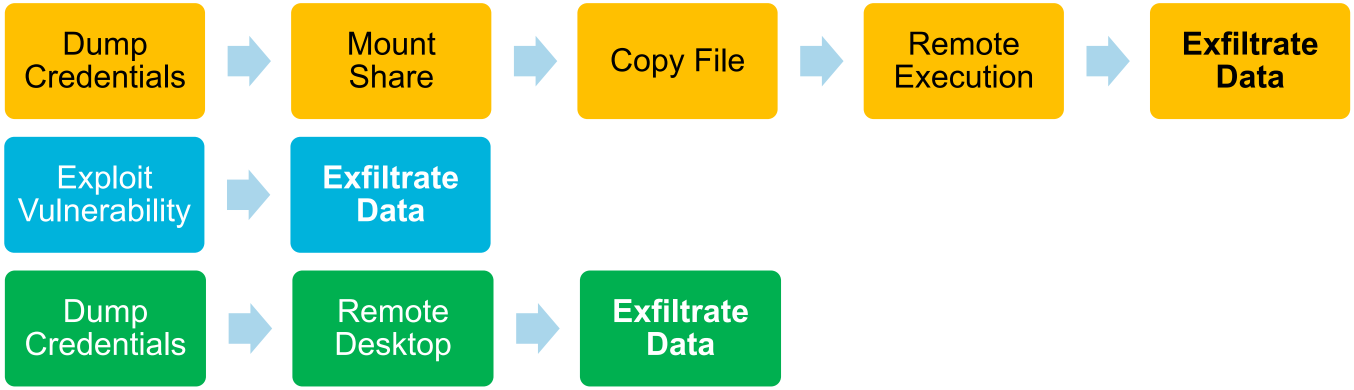


Dump Credentials → Mount Share → Copy File → Remote Execution → **Exfiltrate Data**

Exploit Vulnerability → **Exfiltrate Data**

Building Plans: Remote Desktop Protocol



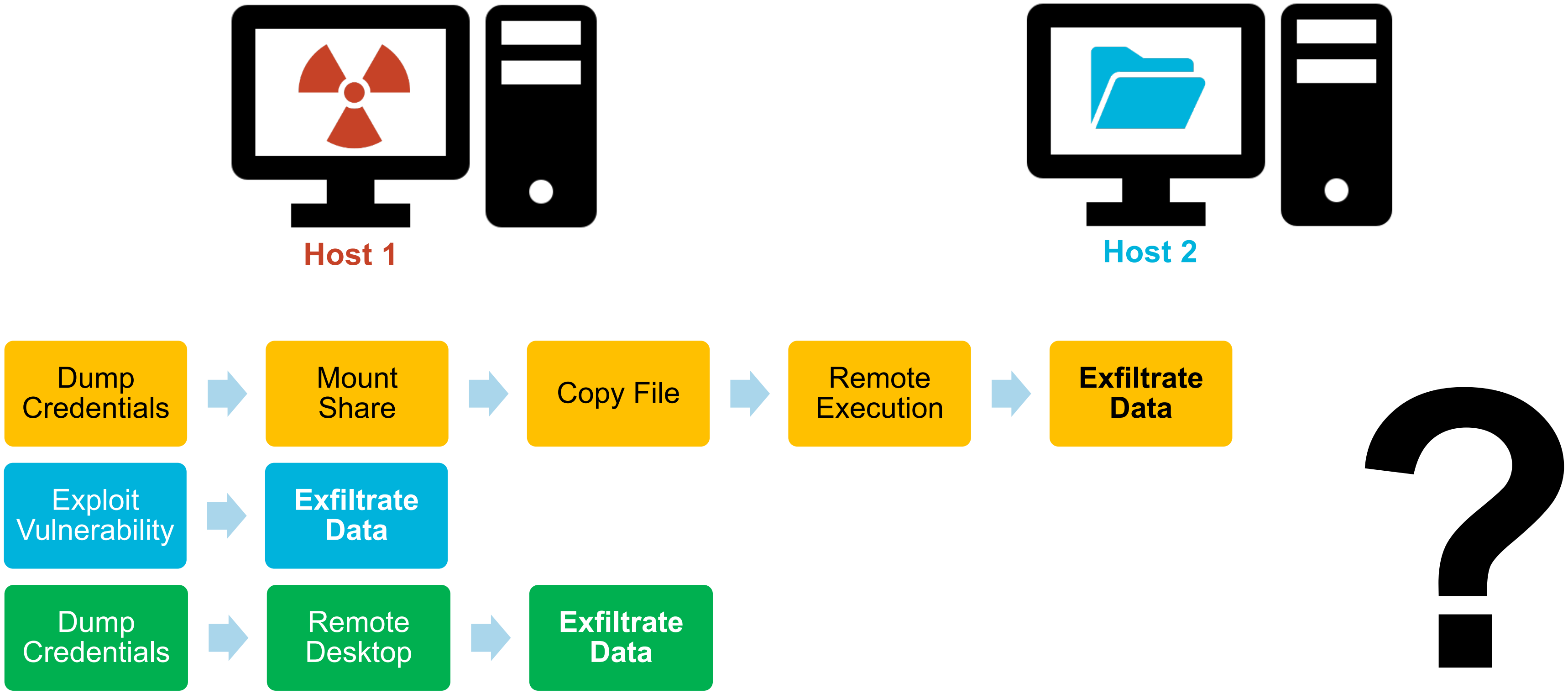


Dump Credentials → Mount Share → Copy File → Remote Execution → **Exfiltrate Data**

Exploit Vulnerability → **Exfiltrate Data**

Dump Credentials → Remote Desktop → **Exfiltrate Data**

Selecting the Right Plan

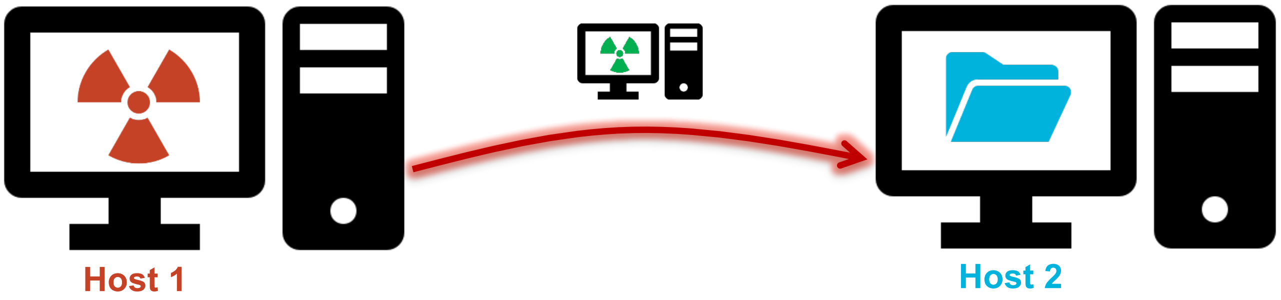


Dump Credentials → Mount Share → Copy File → Remote Execution → **Exfiltrate Data**

Exploit Vulnerability → **Exfiltrate Data**

Dump Credentials → Remote Desktop → **Exfiltrate Data**

Selecting the Right Plan – The CALDERA Heuristic



Dump Credentials → **Mount Share** → **Copy File** → **Remote Execution** → **Exfiltrate Data**

Exploit Vulnerability → **Exfiltrate Data**

Dump Credentials → **Remote Desktop** → **Exfiltrate Data**

$$S(p) = \sum_{i=1}^{n} \frac{R(a_i)}{i}$$

- **Assign each action a *reward***
- **Score plans on summed *weighted rewards***

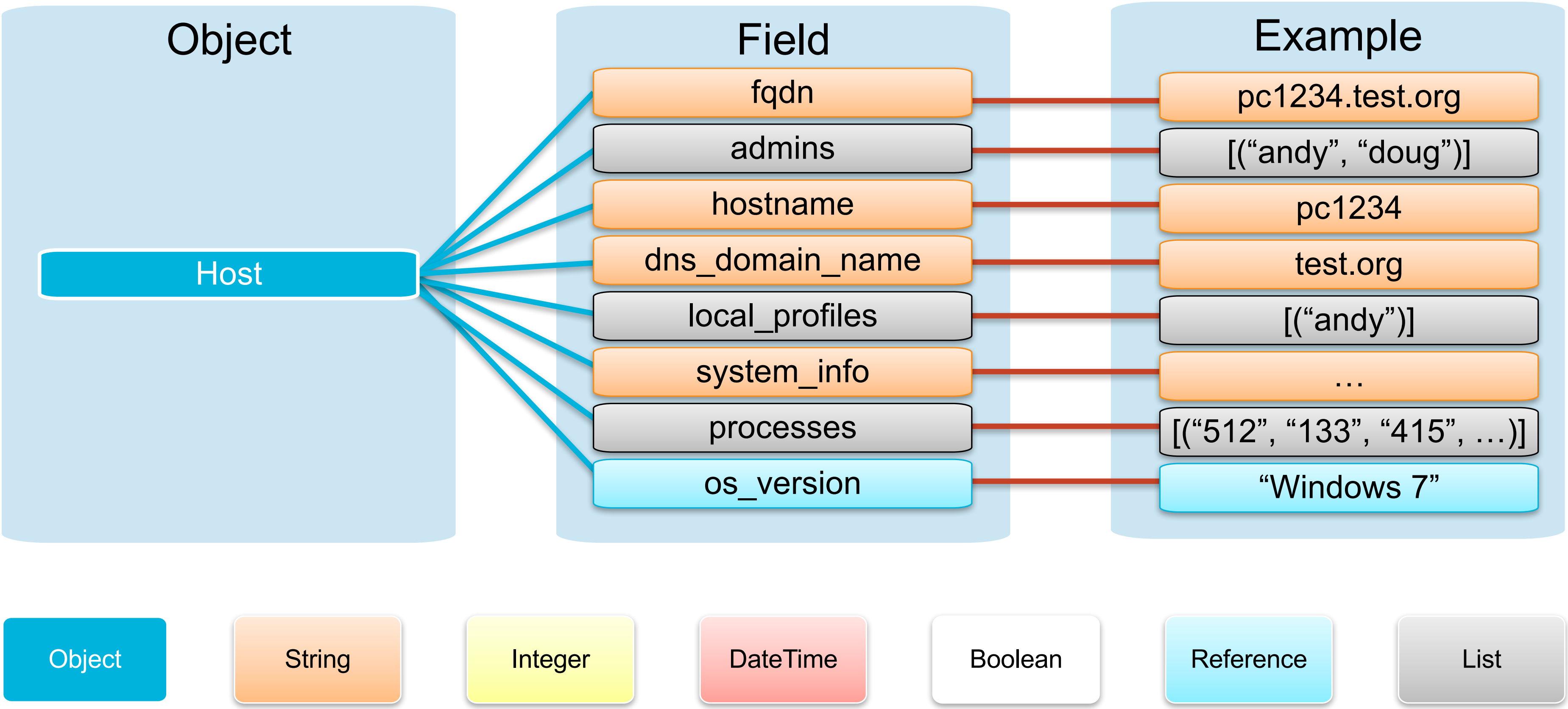
Action	Reward	Action	Reward
Exfiltrate Data	**100**	Copy File	5
Dump Credentials	*50*	Others	1

The Language of Pre/Postconditions: The Data Model

- **Need a way to logically encode what the pre and postconditions of techniques are**
  - Can specify requirements/consequences by specifying facts over a data model

- **CALDERA’s language: *objects* and *typed fields***
  - Objects reference commonly used Windows components
  - Fields specify properties of objects, restricted by type
    - Constructed by default, some fields may not be defined
      - (this is important later!)

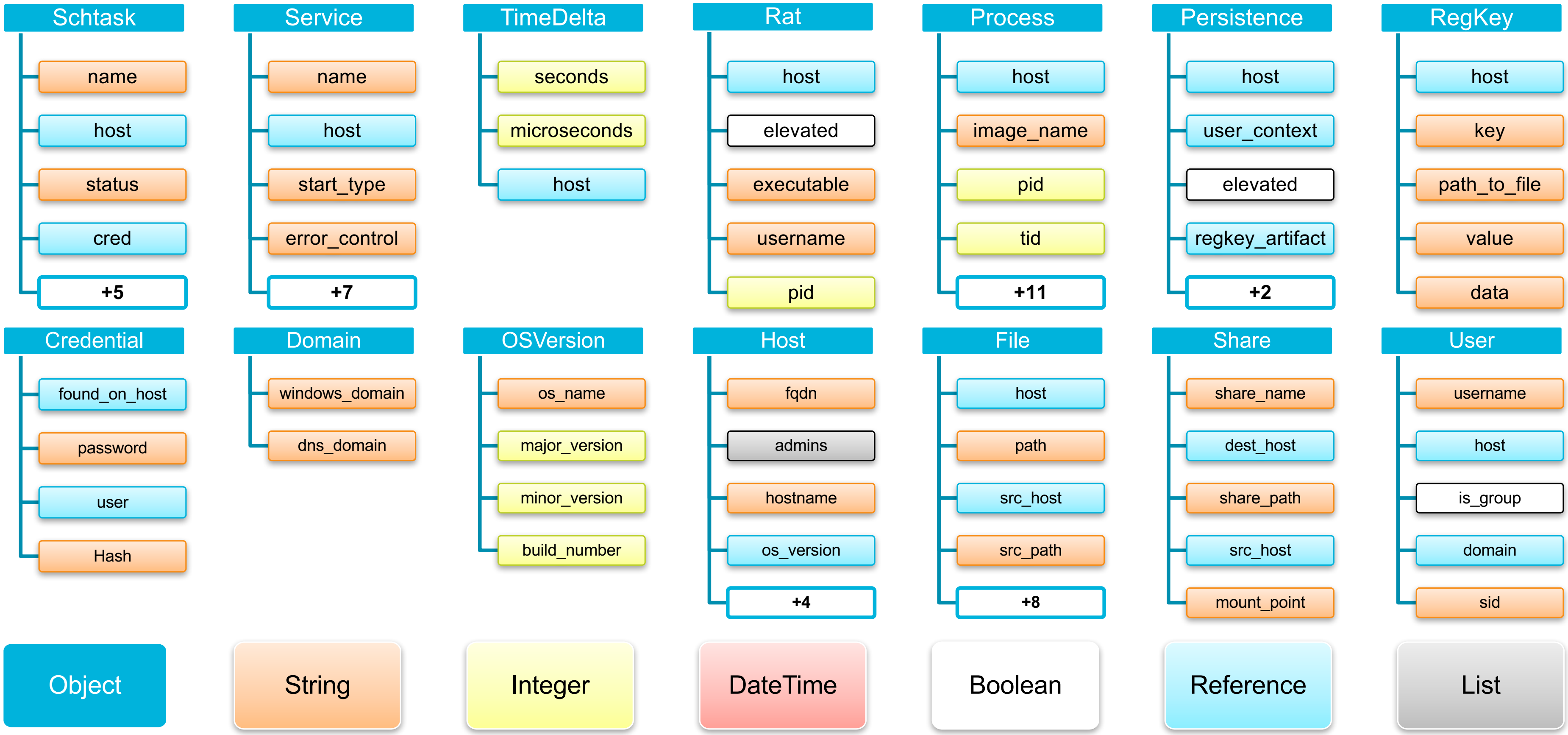
An Example Host Object



Object	Field	Example
Host	fqdn	pc1234.test.org
	admins	[(“andy”, “doug”)]
	hostname	pc1234
	dns_domain_name	test.org
	local_profiles	[(“andy”)]
	system_info	…
	processes	[(“512”, “133”, “415”, …)]
	os_version	“Windows 7”

Object String Integer DateTime Boolean Reference List

Diving into the Data Model



Schtask
- name
- host
- status
- cred
- +5

Service
- name
- host
- start_type
- error_control
- +7

TimeDelta
- seconds
- microseconds
- host

Rat
- host
- elevated
- executable
- username
- pid

Process
- host
- image_name
- pid
- tid
- +11

Persistence
- host
- user_context
- elevated
- regkey_artifact
- +2

RegKey
- host
- key
- path_to_file
- value
- data

Credential
- found_on_host
- password
- user
- Hash

Domain
- windows_domain
- dns_domain

OSVersion
- os_name
- major_version
- minor_version
- build_number

Host
- fqdn
- admins
- hostname
- os_version
- +4

File
- host
- path
- src_host
- src_path
- +8

Share
- share_name
- dest_host
- share_path
- src_host
- mount_point

User
- username
- host
- is_group
- domain
- sid

Object

String

Integer

DateTime

Boolean

Reference

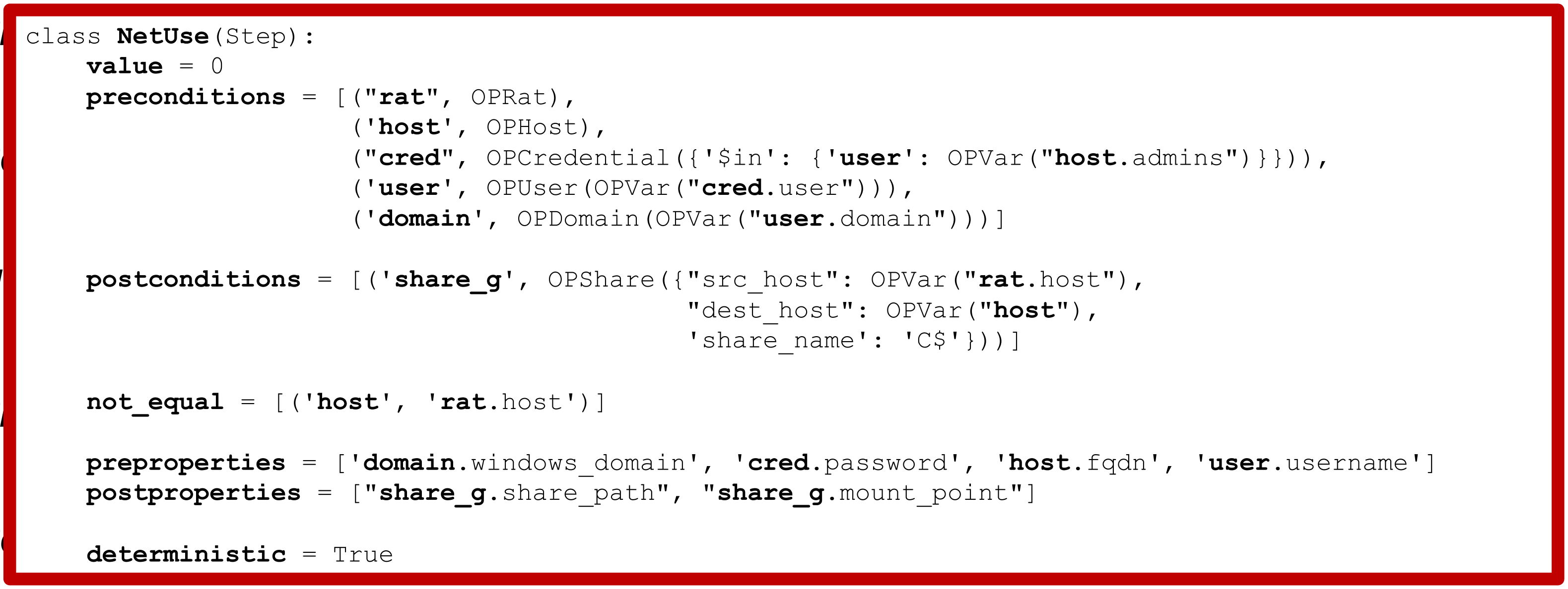
List

Declaring Actions

- **CALDERA provides a syntax to declare actions**
  - ***Preconditions*** specify the requirements
  - ***Postconditions*** specify the effects
  - ***Not_equal*** specifies inequality invariants
  - ***Preproperties*** specify that certain fields must be *defined* but not necessarily a specific value
  - ***Postproperties*** specify that certain fields will be defined after execution

Declaring Actions

- CALDERA provides a syntax to declare actions
  - *Pr*
  - *P(*
  - *N*
  - *Pr* c value
  - *P*

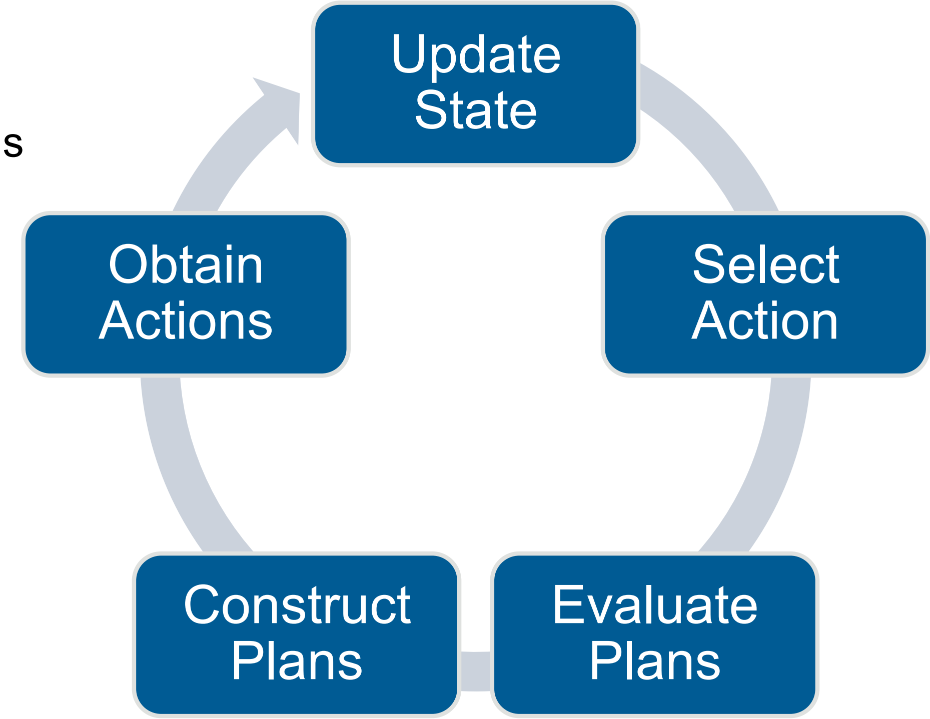


```
class NetUse(Step):
    value = 0
    preconditions = [("rat", OPRat),
                      ('host', OPHost),
                      ("cred", OPCredential({'$in': {'user': OPVar("host.admins")}})),
                      ('user', OPUser(OPVar("cred.user"))),
                      ('domain', OPDomain(OPVar("user.domain")))]

    postconditions = [('share_g', OPShare({"src_host": OPVar("rat.host"),
                                           "dest_host": OPVar("host"),
                                           'share_name': 'C$'}))]

    not_equal = [('host', 'rat.host')]

    preproperties = ['domain.windows_domain', 'cred.password', 'host.fqdn', 'user.username']
    postproperties = ["share_g.share_path", "share_g.mount_point"]

    deterministic = True
```

- Easy, right?

CALDERA's Planning Algorithm

1. Update the world state
2. Figure out all valid actions to execute
3. Construct plans that lead off with those actions
   - Chain actions together by leveraging model
4. Run heuristic to determine *best* plan
5. Execute the first action in the best plan
6. Repeat

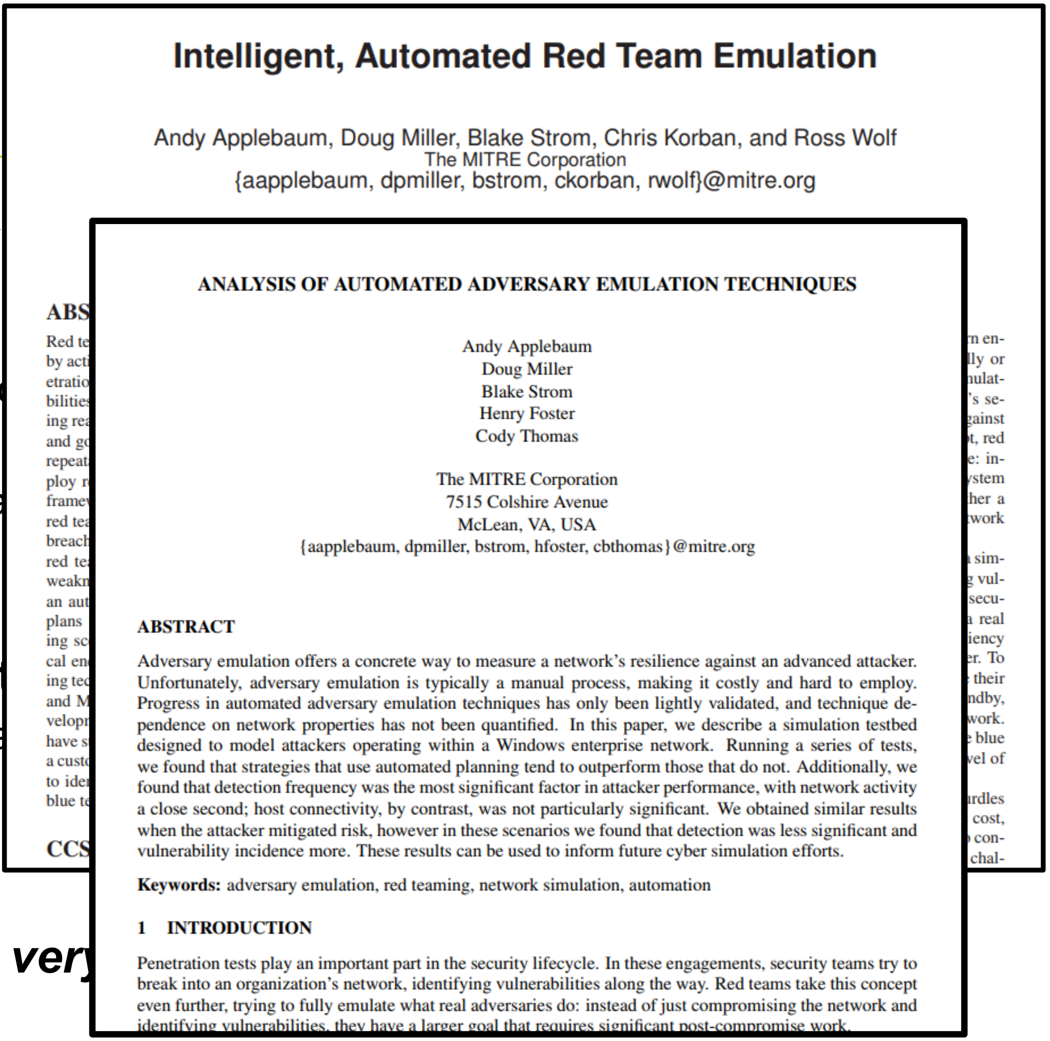


And we’re done! …Right?

- **Using pre and postconditions we can dynamically construct plans and choose actions**
- **Meets our goals:**
  - Can construct plans to make intelligent choices
  - Can easily incorporate new techniques by defining pre/postconditions
  - Can modify rewards/included actions to vary operations
  - Chains weaknesses to achieve goals
  - Functions in new environments*
- **Wait – functions in new environments?**
  - Actually, maybe not: when constructing plans, there is a *significant* amount of uncertainty!
  - Consider dumping credentials:
    - Sometimes they’re *great creds*
    - Sometimes there’s none
- **In reality – handling uncertainty is a *very* hard problem!**

And we’re done! …Right?

- **Using pre and postconditions we ca actions**
- **Meets our goals:**
  - Can construct plans to make intelligent ch
  - Can easily incorporate new techniques by
  - Can modify rewards/included actions to va
  - Chains weaknesses to achieve goals
  - Functions in new environments*
- **Wait – functions in new environment**
  - Actually, maybe not: when constructing pla
  - Consider dumping credentials:
    - Sometimes they’re *great creds*
    - Sometimes there’s none
- **In reality – handling uncertainty is a *very***



Intelligent, Automated Red Team Emulation

Andy Applebaum, Doug Miller, Blake Strom, Chris Korban, and Ross Wolf
The MITRE Corporation
{aapplebaum, dpmiller, bstrom, ckorban, rwolf}@mitre.org

ANALYSIS OF AUTOMATED ADVERSARY EMULATION TECHNIQUES

Andy Applebaum
Doug Miller
Blake Strom
Henry Foster
Cody Thomas

The MITRE Corporation
7515 Colshire Avenue
McLean, VA, USA
{aapplebaum, dpmiller, bstrom, hfoster, cbthomas}@mitre.org

ABSTRACT

Adversary emulation offers a concrete way to measure a network’s resilience against an advanced attacker. Unfortunately, adversary emulation is typically a manual process, making it costly and hard to employ. Progress in automated adversary emulation techniques has only been lightly validated, and technique dependence on network properties has not been quantified. In this paper, we describe a simulation testbed designed to model attackers operating within a Windows enterprise network. Running a series of tests, we found that strategies that use automated planning tend to outperform those that do not. Additionally, we found that detection frequency was the most significant factor in attacker performance, with network activity a close second; host connectivity, by contrast, was not particularly significant. We obtained similar results when the attacker mitigated risk, however in these scenarios we found that detection was less significant and vulnerability incidence more. These results can be used to inform future cyber simulation efforts.

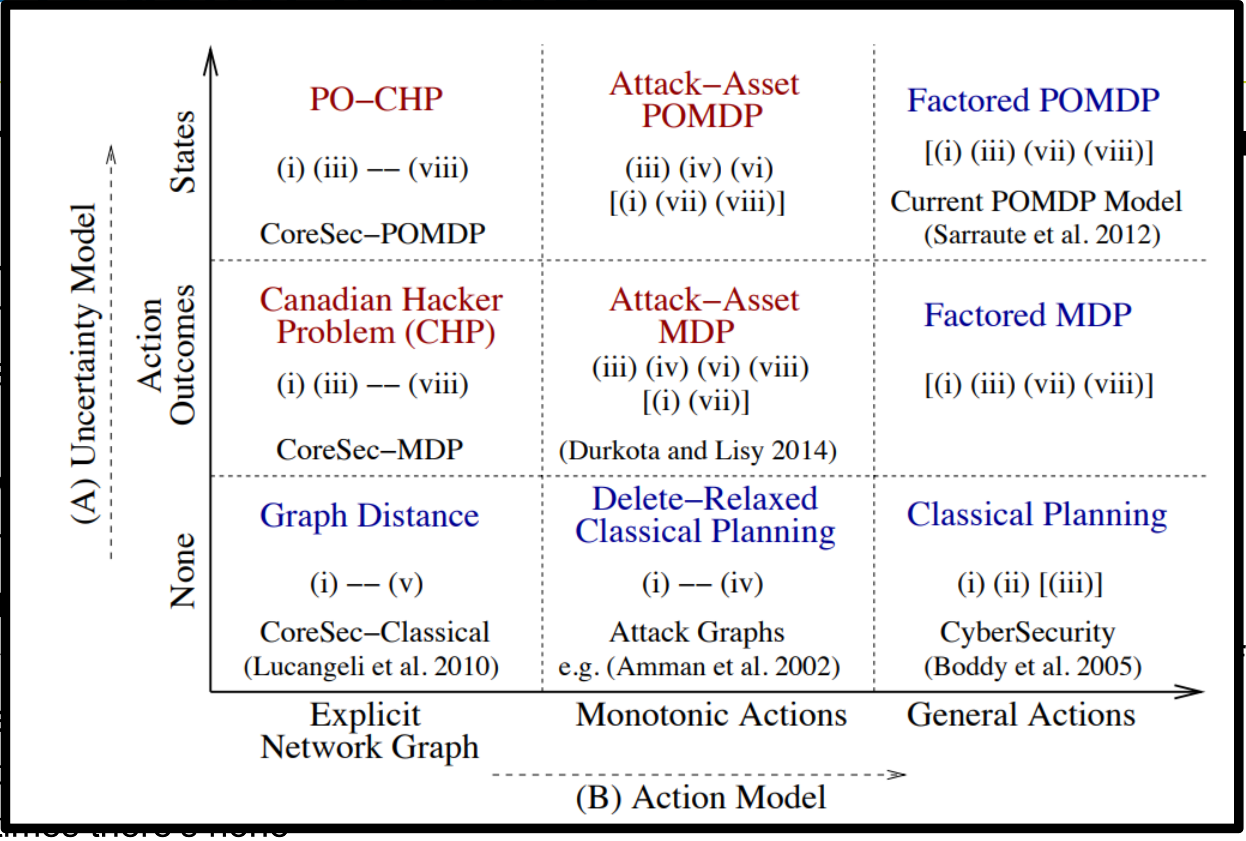
Keywords: adversary emulation, red teaming, network simulation, automation

1 INTRODUCTION

Penetration tests play an important part in the security lifecycle. In these engagements, security teams try to break into an organization’s network, identifying vulnerabilities along the way. Red teams take this concept even further, trying to fully emulate what real adversaries do: instead of just compromising the network and identifying vulnerabilities, they have a larger goal that requires significant post-compromise work.

And we're done! …Right?

- **Using pr** **ns and choose actions**
- **Meets ou**
  - Can cor
  - Can eas
  - Can mo
  - Chains v
  - Functior
- **Wait – fu**
  - Actually f uncertainty!
  - Conside
    - Somet
    - Sometimes there's none

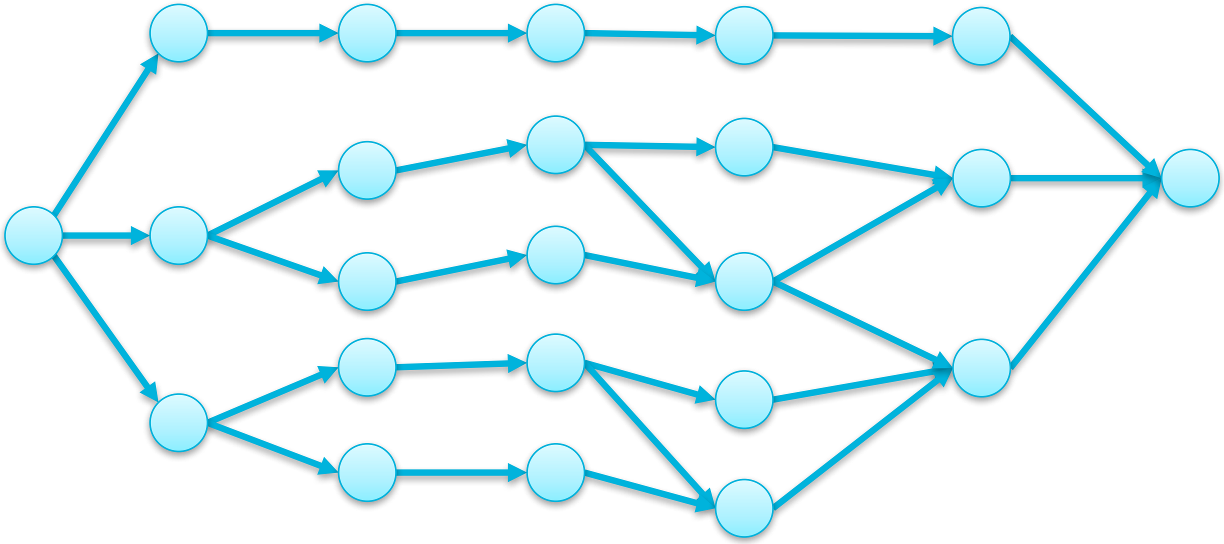


	Explicit Network Graph	Monotonic Actions	General Actions
States	PO–CHP (i) (iii) –– (viii) CoreSec–POMDP	Attack–Asset POMDP (iii) (iv) (vi) [(i) (vii) (viii)]	Factored POMDP [(i) (iii) (vii) (viii)] Current POMDP Model (Sarraute et al. 2012)
Action Outcomes	Canadian Hacker Problem (CHP) (i) (iii) –– (viii) CoreSec–MDP	Attack–Asset MDP (iii) (iv) (vi) (viii) [(i) (vii)] (Durkota and Lisy 2014)	Factored MDP [(i) (iii) (vii) (viii)]
None	Graph Distance (i) –– (v) CoreSec–Classical (Lucangeli et al. 2010)	Delete–Relaxed Classical Planning (i) –– (iv) Attack Graphs e.g. (Amman et al. 2002)	Classical Planning (i) (ii) [(iii)] CyberSecurity (Boddy et al. 2005)

- **Hoffmann, Jörg. "Simulated Penetration Testing: From" Dijkstra" to" Turing Test++"." *ICAPS*. 2015.**

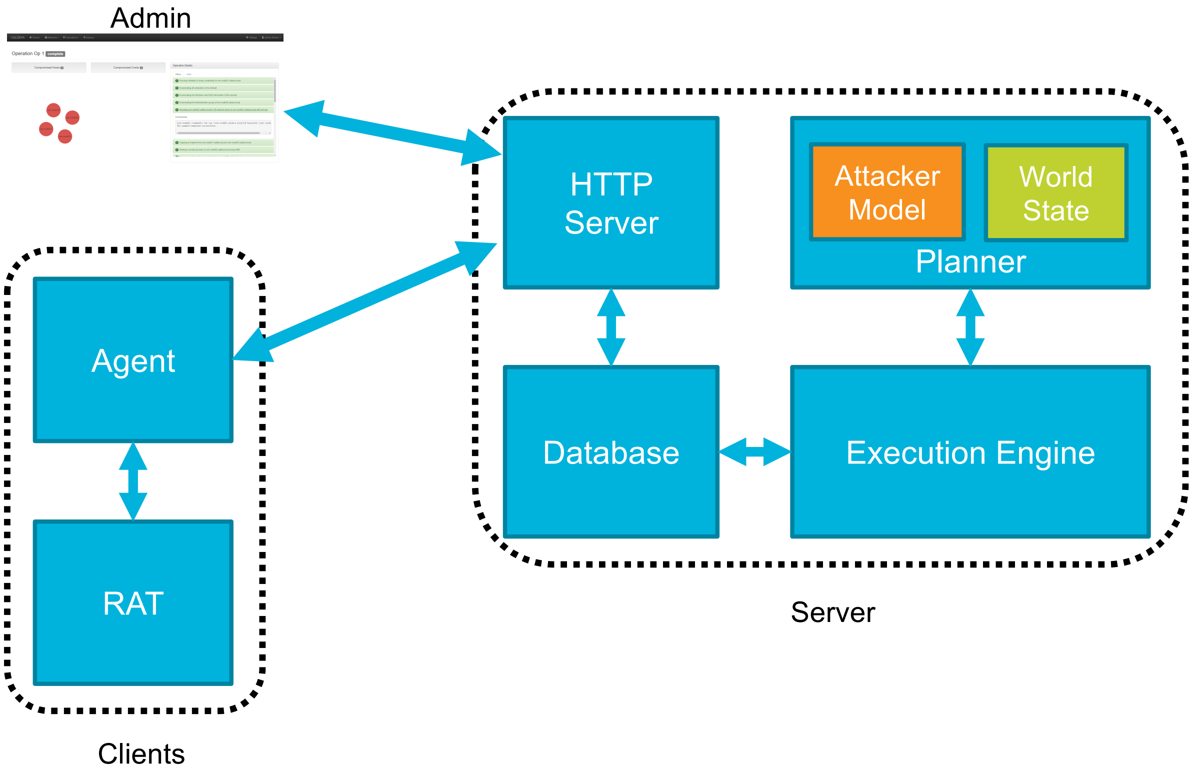
A Quick Fix with Hints

- **If we can't predict the outcome of an action, use hints**
- **Hints are crafted to be the "best" outcome of the action**
  - E.g. performing credential dumping gives me a "useful" credential



Architecture

System Architecture

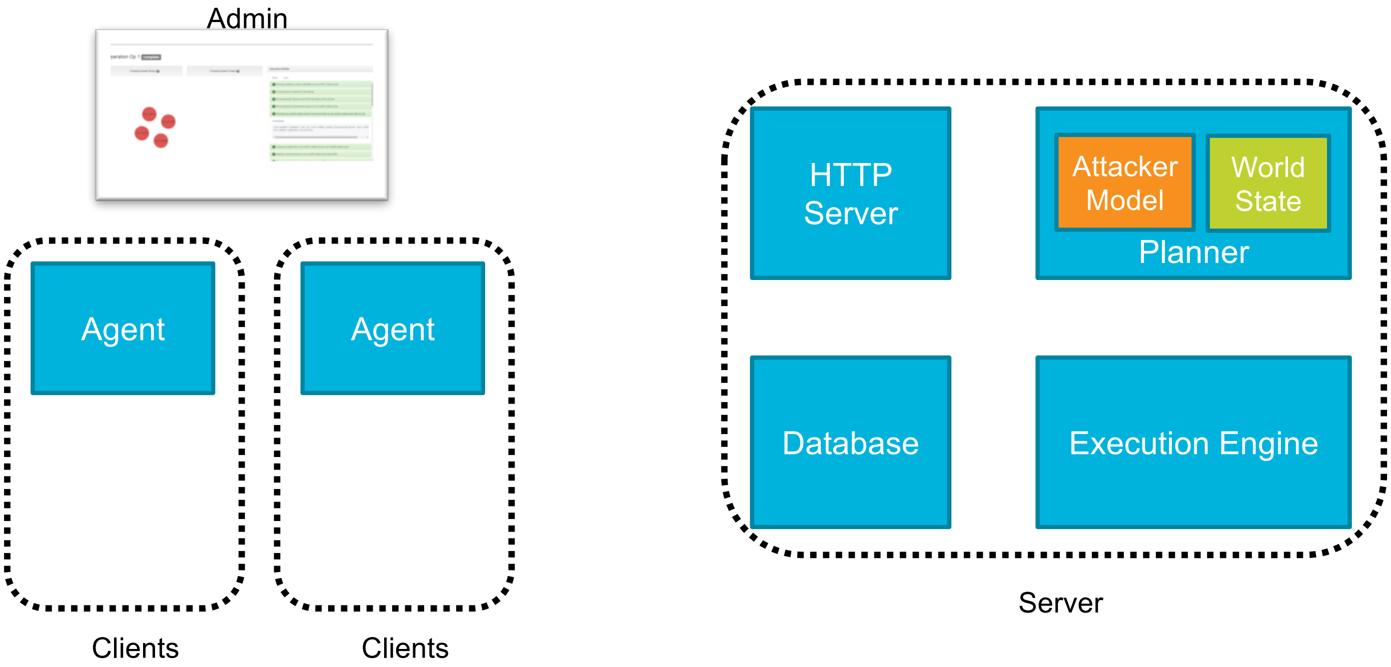


System Architecture

- Server and Agent written in Python 3
- Rat written in C#
- MongoDB
- Web interface is a JavaScript based web app
- pyDatalog logic backend

Demo

Walking Through an Operation



Admin

Agent

Agent

Clients

Clients

HTTP Server

Attacker Model

World State

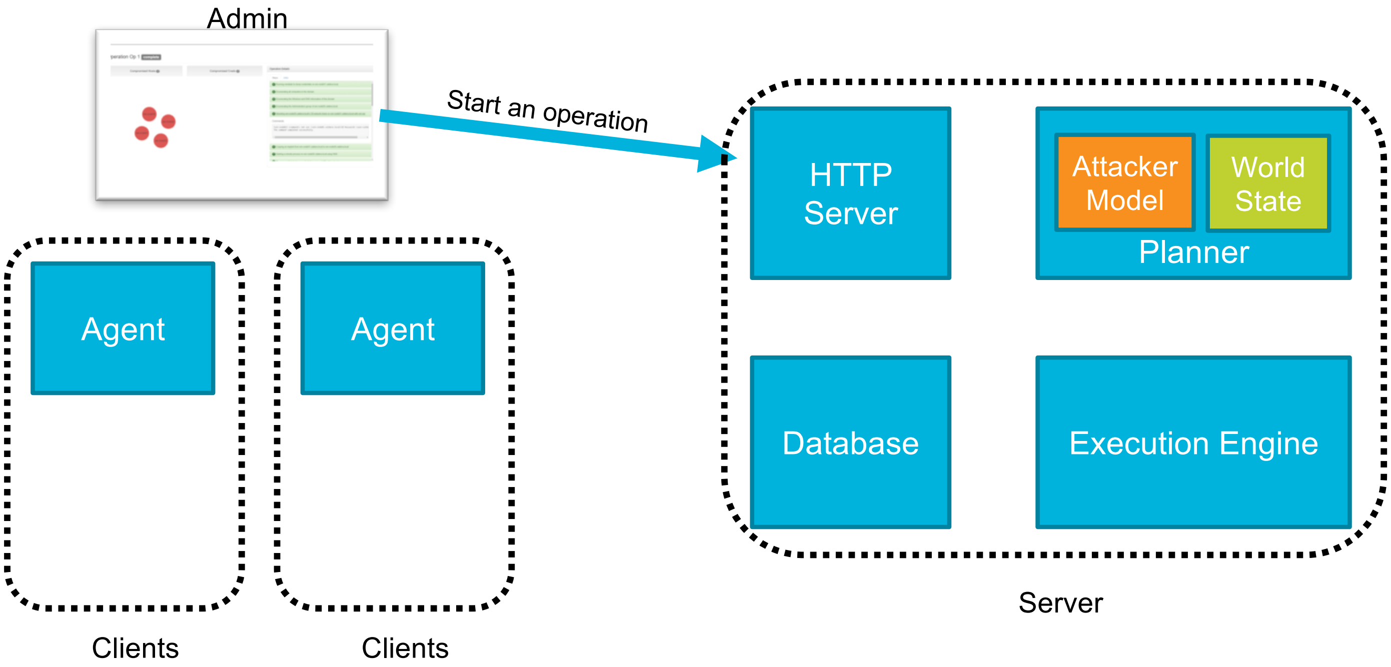
Planner

Database

Execution Engine

Server

Walking Through an Operation



Admin

Start an operation

HTTP Server

Attacker Model

World State

Planner

Database

Execution Engine

Server

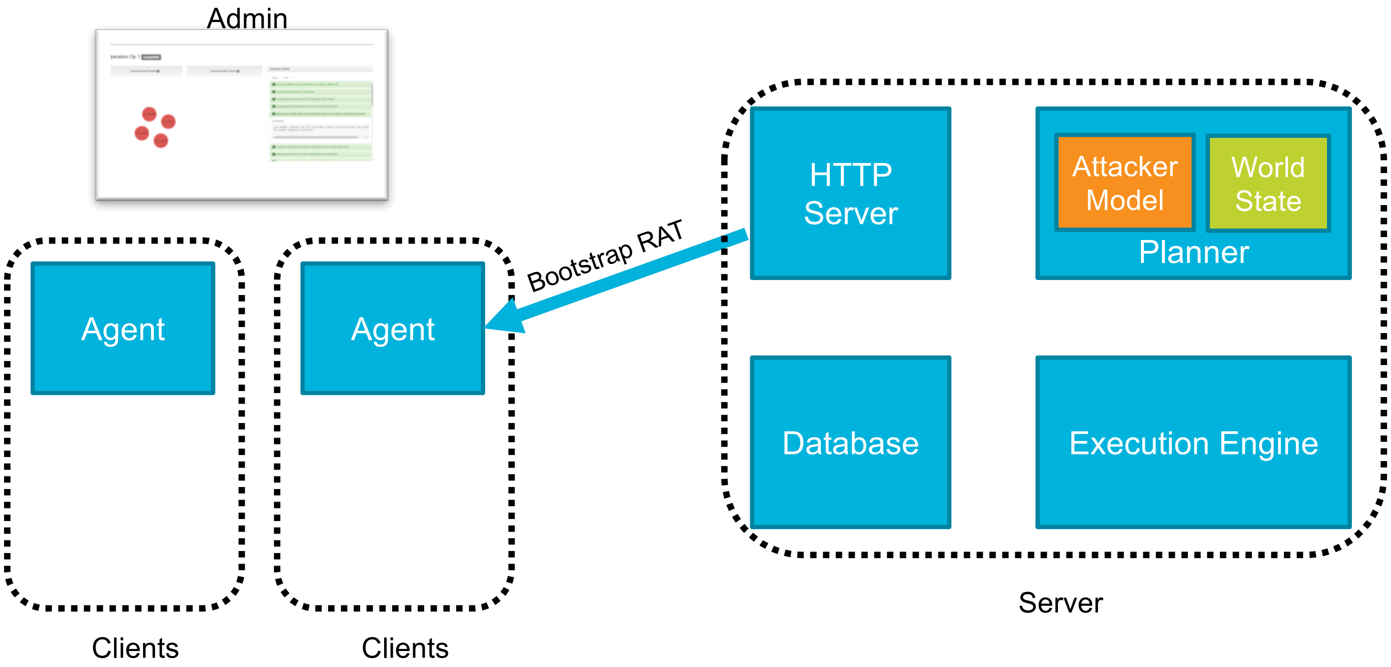
Agent

Agent

Clients

Clients

Walking Through an Operation



Admin

Agent

Agent

Clients

Clients

Bootstrap RAT

HTTP Server

Attacker Model

World State

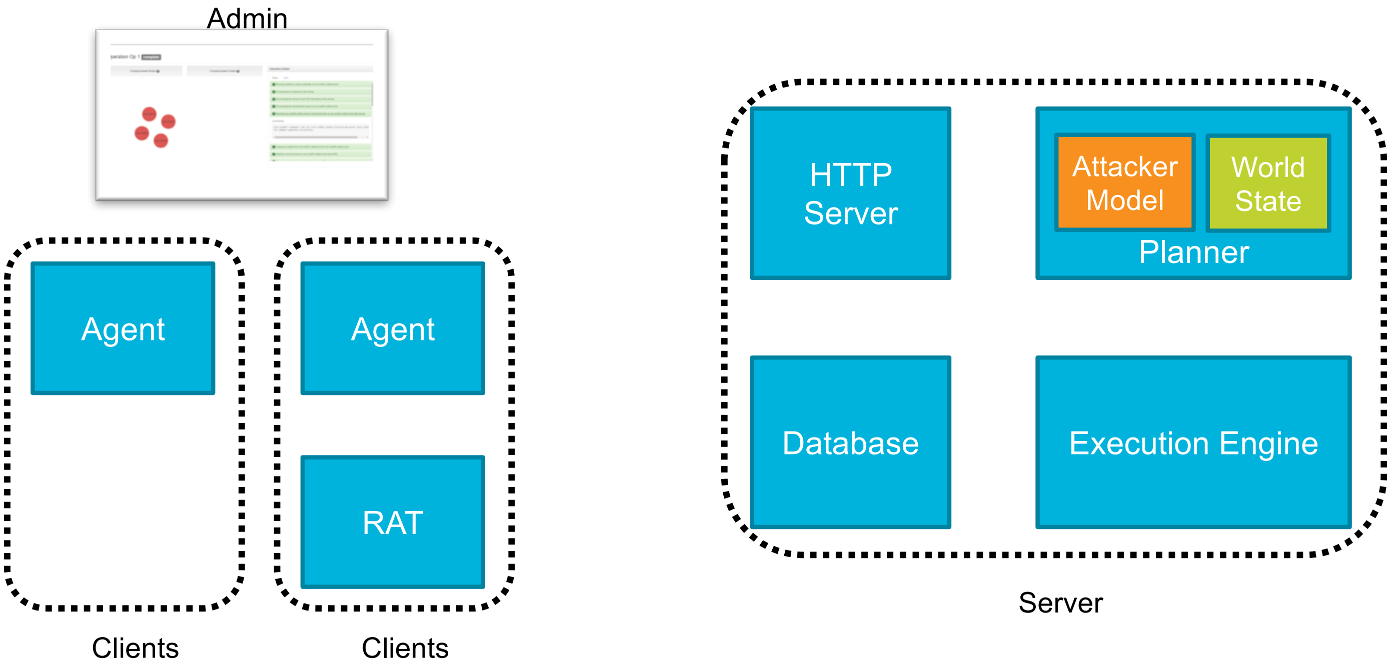
Planner

Database

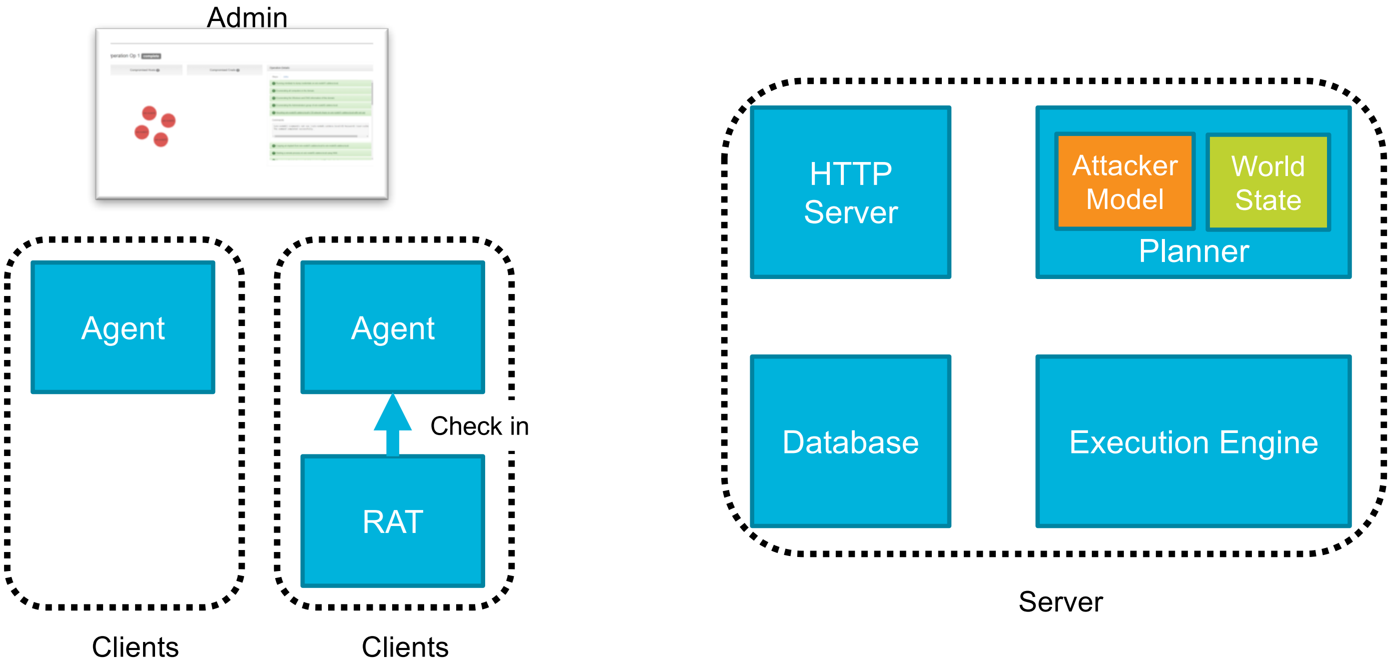
Execution Engine

Server

Walking Through an Operation



Walking Through an Operation



Admin

Agent

Clients

Agent

Check in

RAT

Clients

HTTP Server

Attacker Model

World State

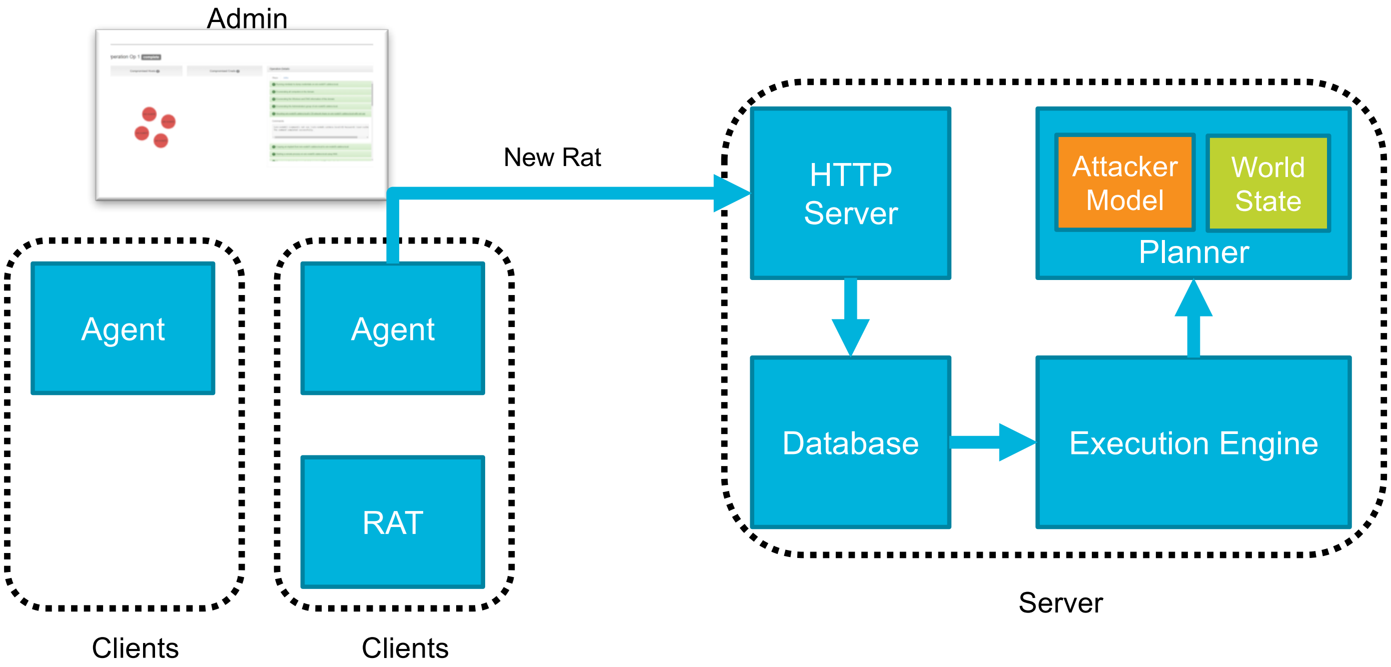
Planner

Database

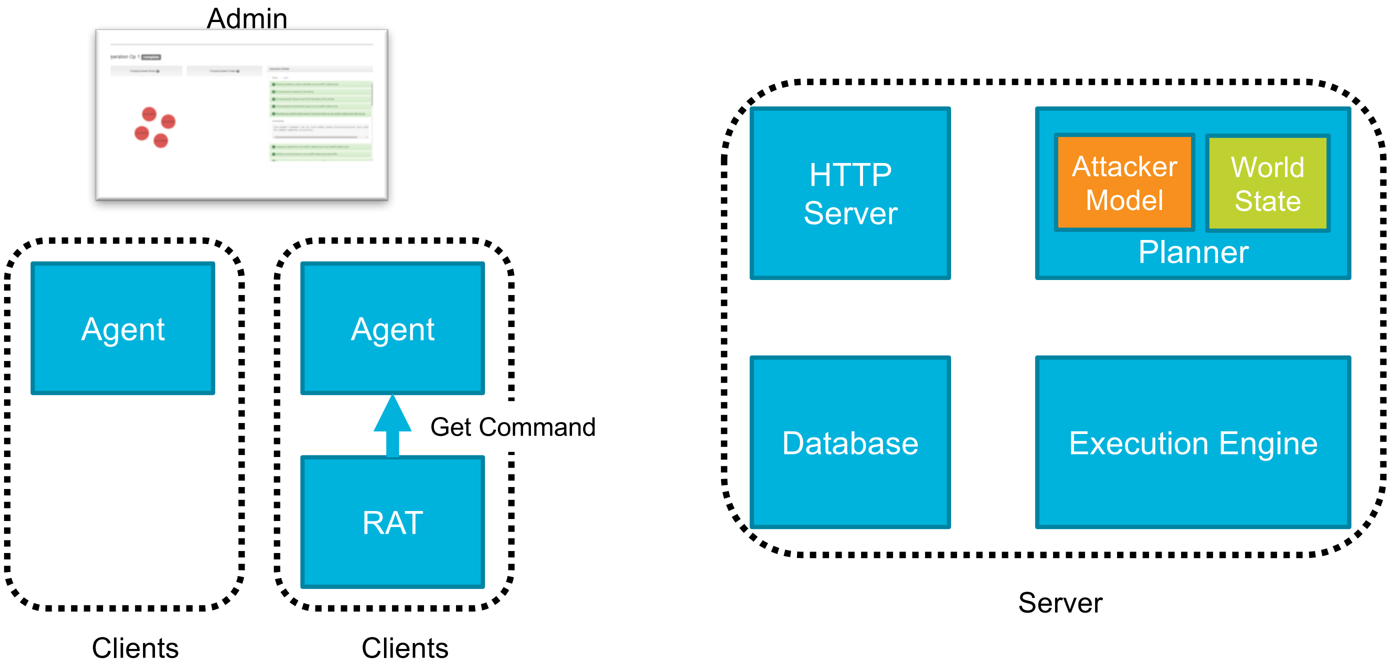
Execution Engine

Server

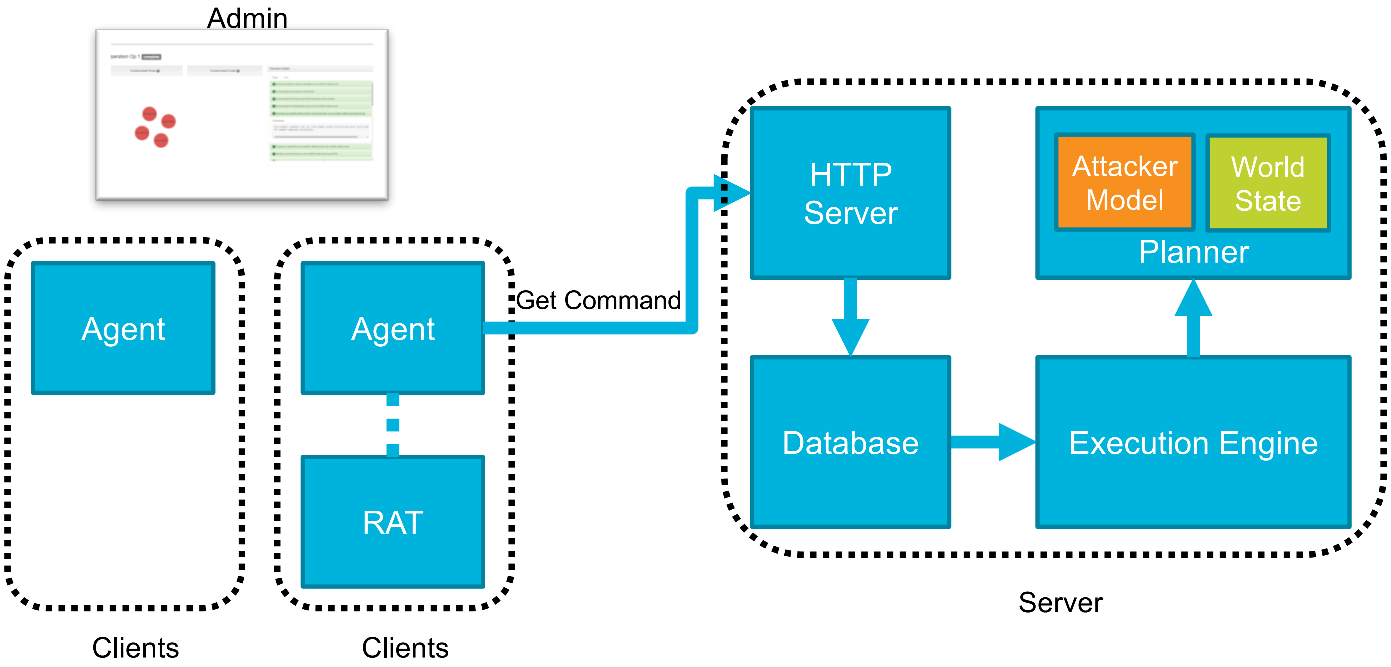
Walking Through an Operation



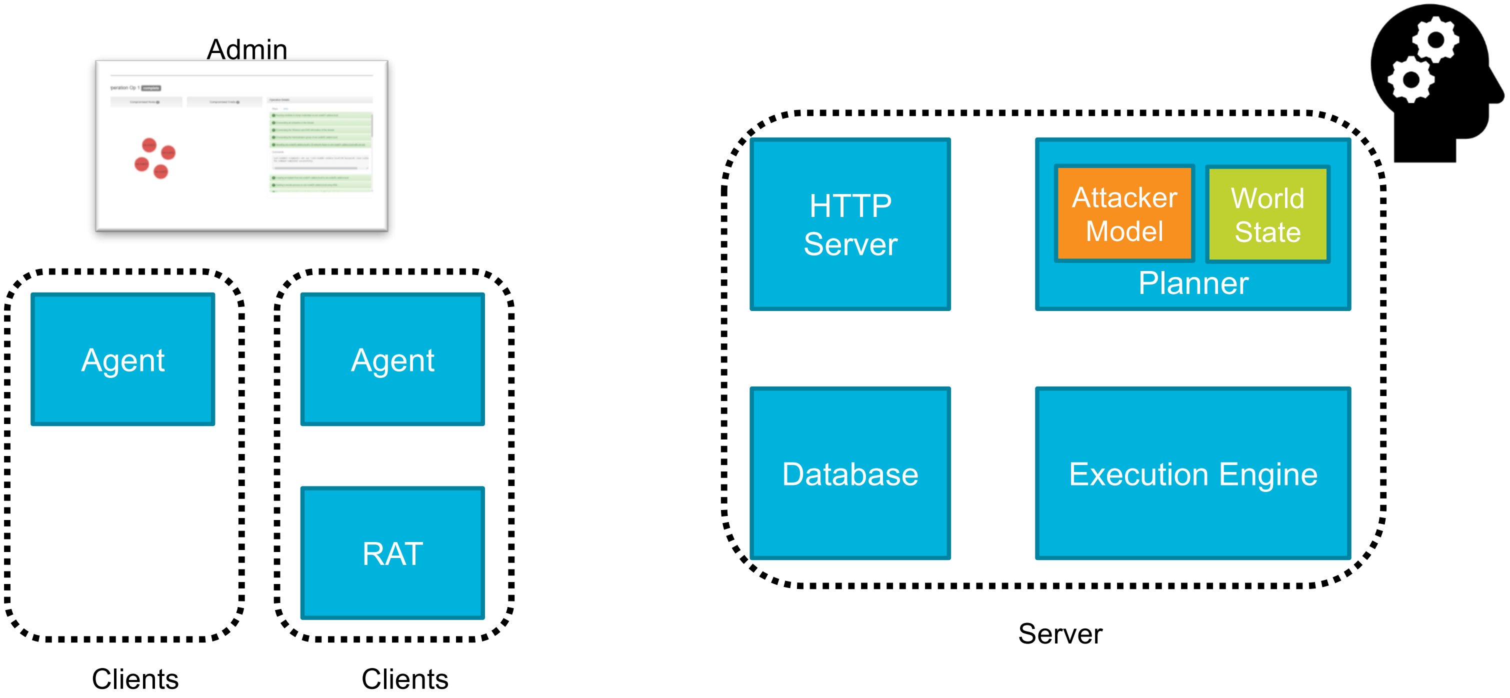
Walking Through an Operation



Walking Through an Operation



Walking Through an Operation



Admin

Agent

Clients

Agent

RAT

Clients

HTTP Server

Attacker Model

World State

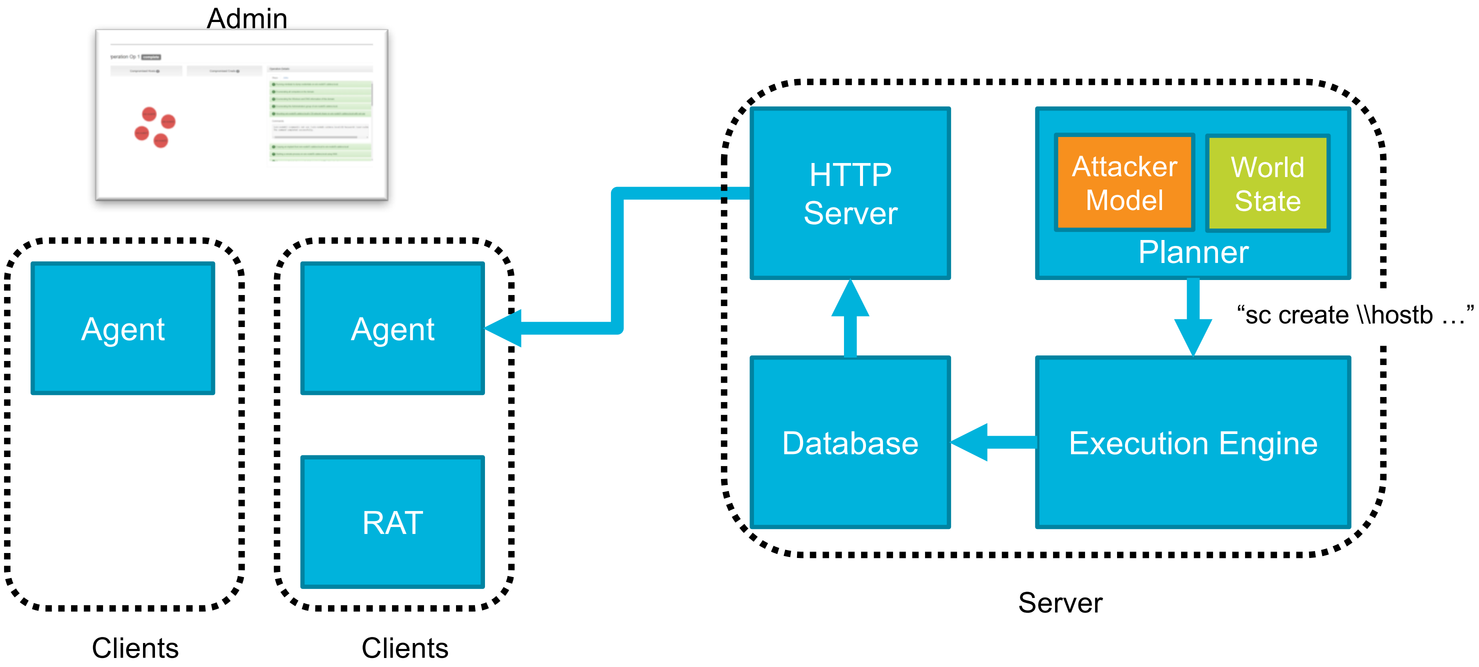
Planner

Database

Execution Engine

Server

Walking Through an Operation



Admin

Agent

Clients

Agent

RAT

Clients

HTTP Server

Attacker Model

World State

Planner

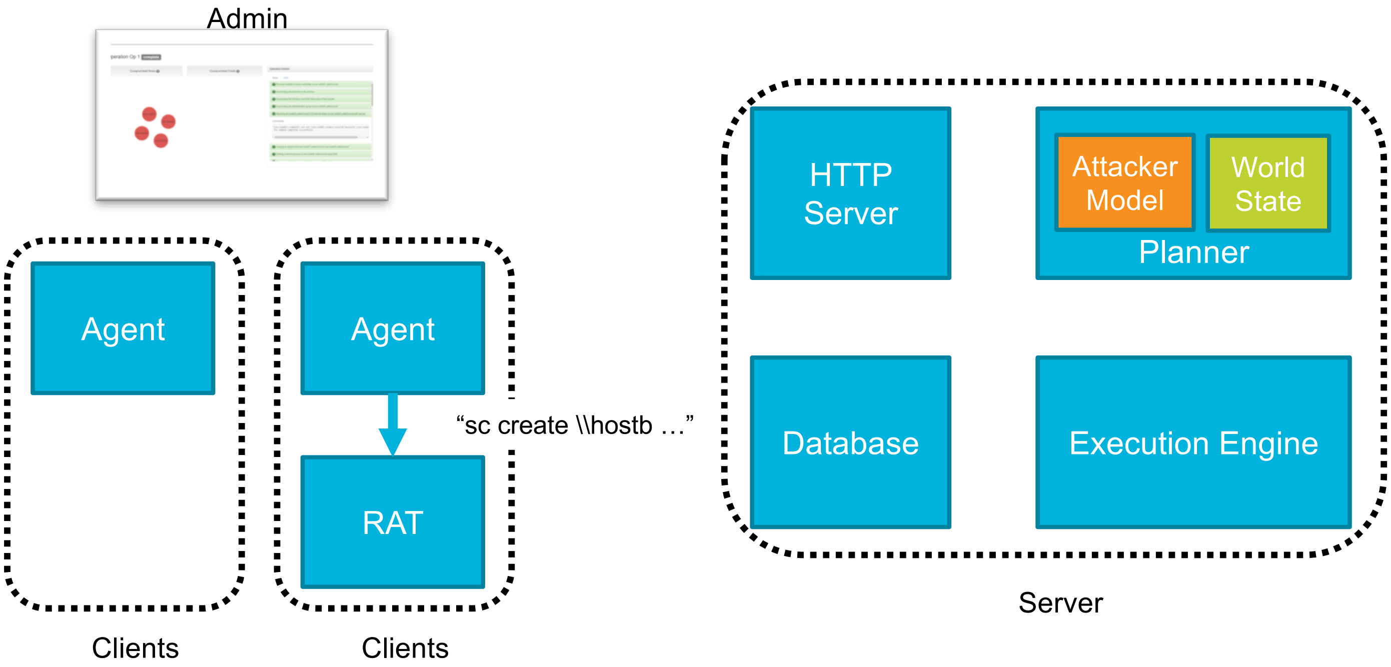
"sc create \\hostb …"

Database

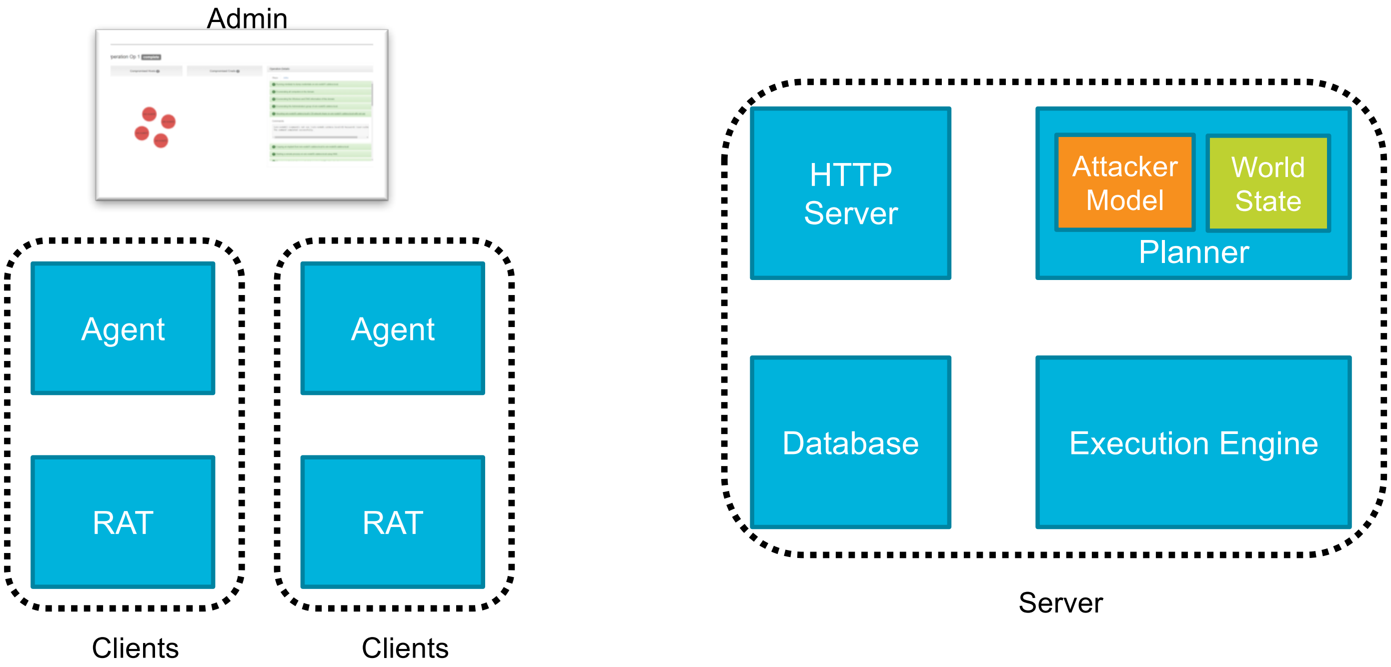
Execution Engine

Server

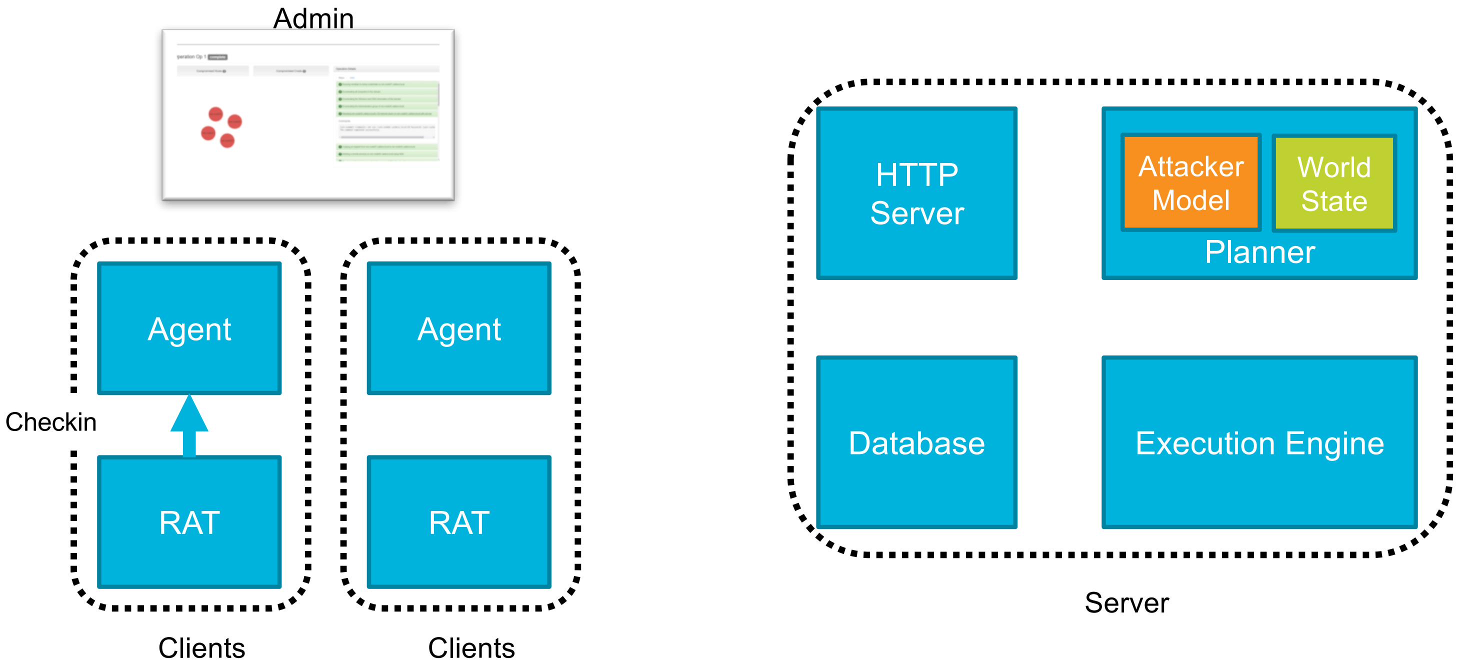
Walking Through an Operation



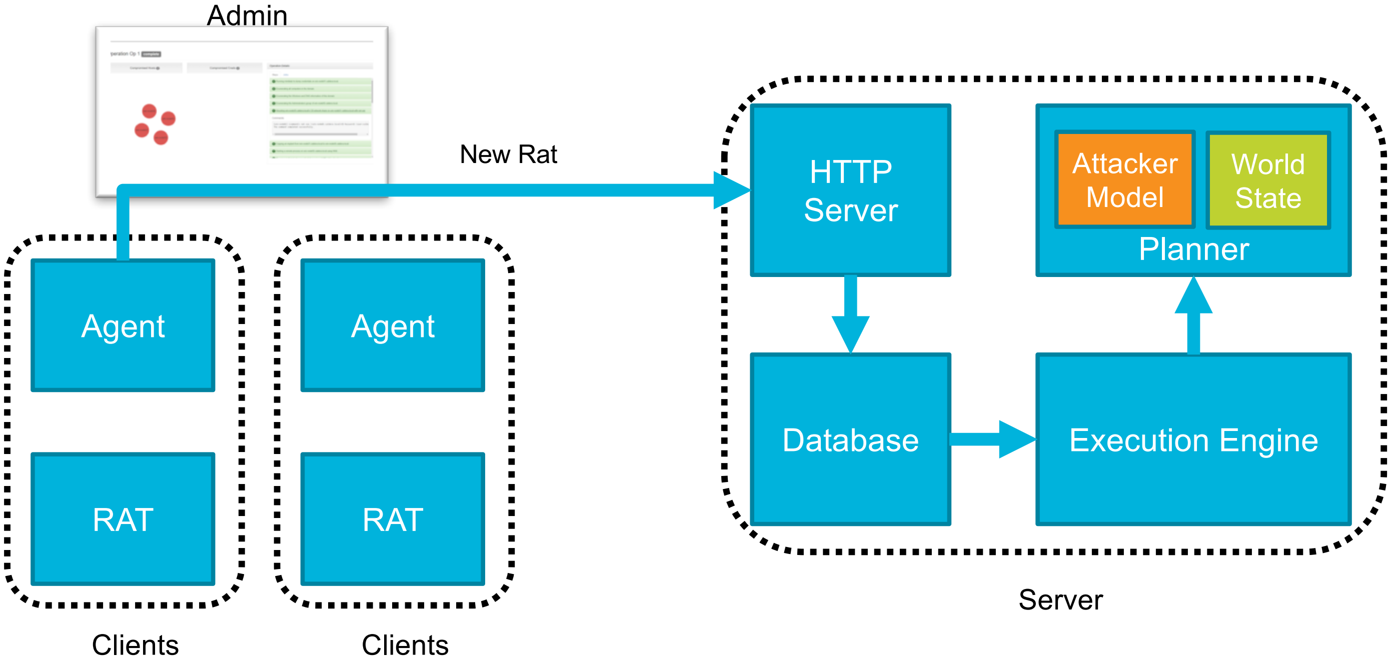
Walking Through an Operation



Walking Through an Operation



Walking Through an Operation



CALDERA Applications

- Testing analytics and sensors – Does my stuff work?
- Data generation – What does bad look like?
- Red/blue team training – I need practice!

Community Participation

- **Want to get involved with CALDERA? We accept Pull Requests**
  - Bugfixes
  - Implement a new adversary (ATT&CK) technique
  - Usability features
  - Integration with other tools and frameworks
  - Enhancements to our data model

Other (Free) Tools

BloodHound – Attack Path Generation

- https://github.com/BloodHoundAD/BloodHound

GoFetch – Automatic Execution of BloodHound paths with PowerShell tools

- https://github.com/GoFetchAD/GoFetch

ANGRYPUPPY – Automatic execution of BloodHound paths with Cobalt Strike

- https://github.com/vysec/ANGRYPUPPY

Death Star – Automatic Execution of attack paths with PowerShell Empire

- https://github.com/byt3bl33d3r/DeathStar

Atomic Red Team

- https://github.com/redcanaryco/atomic-red-team

Metta

- <no url yet>

(Probably more, sorry if we missed you)

Related (MITRE) Efforts

- **BRAWL: Automated Bot-vs-Bot Games**
  - Free data!
  - https://github.com/mitre/brawl-public-game-001
- **BRAWL Shared Format (BSF)**
  - Standardized format to correlate red bot vs blue bot cyber games
- **CASCADE: Automated Host-based Investigations**
  - https://github.com/mitre/cascade-server

Why this Matters

- The False Negative problem is real
- Offensive testing with Adversary Emulation can help
- Automation (like CALDERA) and human adversary emulation are complementary
- Pre and postconditions + planning are powerful
- Help Us!

Show me the code!

github.com/mitre/caldera

MITRE

MITRE is a not-for-profit organization whose sole focus is to operate federally funded research and development centers, or FFRDCs. Independent and objective, we take on some of our nation's—and the world's—most critical challenges and provide innovative, practical solutions.

Learn and share more about MITRE, FFRDCs, and our unique value at www.mitre.org