CRADLE: A Language for Cyber Event Reproduction and System-wide Analysis

Zhenkai Liang

NUS National University of Singapore | Computing

National Cybersecurity R&D Laboratory

Missing Details in Cyber Attack Reports

- CTI reports only consist brief information about an incident
- Researchers & analysts require technical fine-grained details
  - Game/movie vs. summary

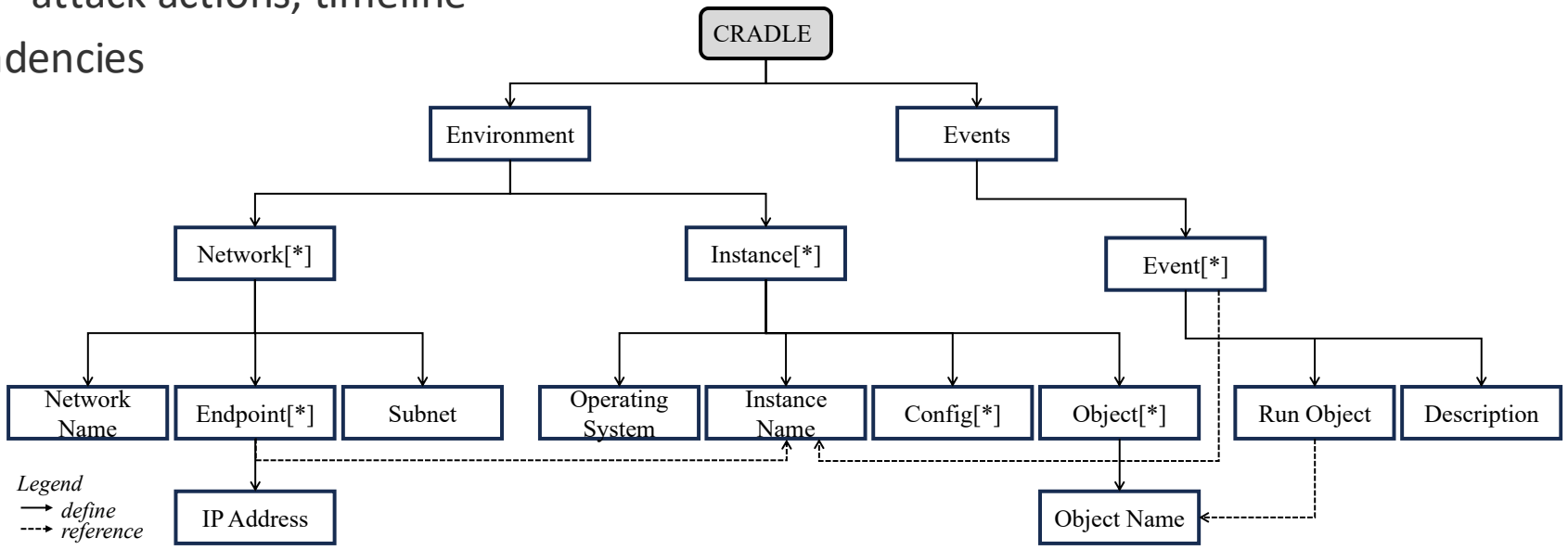
CRADLE Language

CRADLE: Language of Cyber Experimentation

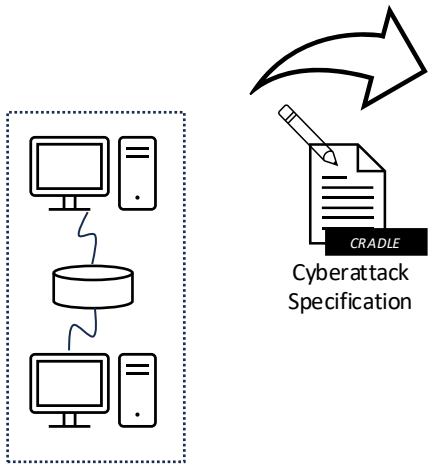
Cyber-event **R**econstruction & **A**utomation **D**escription **L**anguag**e** (CRADLE)

- High-level domain-specific language to define:
  - Environment – instances, networks, artifacts, configurations
  - Event – attack actions, timeline
  - Dependencies

Environment $E = (I, N, O, Ev)$ where:
Instance $I_i = (Name, OS, Objects, Role)$
Network $N_i = (Name, Subnet, Endpoints)$
Object $O_i = (Name, URI)$
Event $Ev_i = (Name, Instance, runObject, Schedule, Description, ..)$



CRADLE Example



APT17 CRADLE Specification

```
instances() >
   instance("router"),
   instance("target"),
   instance("attacker").

instance("router") >
   os("ubuntu", "20.04").

instance("attacker") >
   os("ubuntu", "20.04"),
   object("exploitpy"),
   config("sphere-linux-python3"),
   config("sphere-linux-sliver-server"),
   config("sphere-linux-auto_deploy"),
   config("sphere-linux-zer0cool"),
   config("sphere-deploy-config"),
   config("sphere-modification-sliver"),
   config("sphere-linux-auditd"),
   config("sphere-linux-tcpdump").
```

```
networks() >
   network("ExternalNetwork"),
   network("InternalNetwork").

network("ExternalNetwork") >
   subnet("192.168.56.0/24"),
   endpoint("router", "192.168.56.10"),
   endpoint("attacker", "192.168.56.100"),
   endpoint("target", "192.168.56.200").
```

```
mainEvent() >
   event("1").

event("1") >
   instance("attacker"),
   needRoot("true"),
   subject("bash", ""),
   runObject("exploitpy", ""),
   pauseBeforeRun("0"),
   pauseAfterRun("5"),
   waitfor("false"),
   scheduleExecution("2024-09-15T08:30:00Z"),
   description("Run attack chain").
```

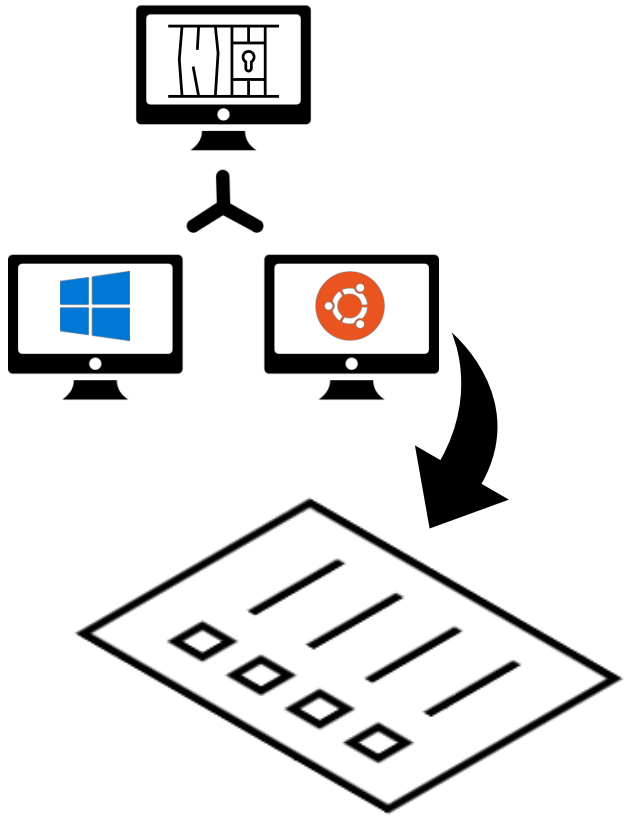
Cyber Incident Constructs

Construct	Purpose
`event(...)`	Defines a single attack step
`instance(...)`	Represents a system or host in the environment
`runObject(...)`	Declares the action/tool to execute
`object(...)`	Declares reusable scripts/tools and their locations
`needRoot(...)`	Models execution privilege
`subject(...)`	Defines the shell/interpreter context (e.g., bash, pwsh)
`pauseBeforeRun, pauseAfterRun`	Introduce timing control
`waitfor(...)`	Models control flow dependencies between events
`network(...), endpoint(...)`	Define network-aware segmentation and visibility
`config(...)`	System software pre-conditions or monitoring instrumentation
`description(...)`	Human-readable intent, essential for semantics traceability

Cyber Incident Semantics

Semantic Concept	What It Captures	CRADLE Example
Attack Phase (pre/main/post)	Structure of attack lifecycle	`preEvent(), mainEvent(), postEvent()`
Action Representation	Adversarial actions as executable objects	`runObject("mimikatz-execute")`
Actor-Target Assignment	Binding an action to a specific host	`instance("VictimMachine") in event(...)`
Privilege Semantics	Root vs. user execution context	`needRoot("true")`
Temporal Semantics	Event scheduling and timing	`scheduleExecution(...), pauseBeforeRun`
Dependency Semantics	Event dependencies and ordering	`waitfor("event(2)")`
Network-Aware Context	Network topology affecting attack reachability	`endpoint(...) within a network(...)`
Behavioral Intent	Explicit attacker intent in descriptions	`description("Establish C2 backdoor")`
Artifact Traceability	Reuse of tools/scripts with versioned URLs	`location("${uriRemote}/...")`

Cyber Experimentation as Code (CEaC)



Encoding Cyber Attack As Code

- **Reproducibility and Repeatability**
  - Current simulations are manual and single-use
- **Capture the cyberattack context and semantics**
  - Codify the cyberattack as a portable specification
- **Coding Agility**
  - Enable *versioning* of attack scenarios to keep pace with evolving APTs

- Existing '***X-as-Code***' paradigms handle the provisioning & configuration, but lack the ability to capture the narrative of a cyberattack
  - *"assembly programming"* for experimentation

RECAP

End-to-end Dataset Generation Framework

Data For Cyberattack Analysis

Need For Quality Data

Modern threat detection and root cause analysis rely heavily on data-driven models, creating a critical dependency on high-fidelity training data

Limited Availability Of Data

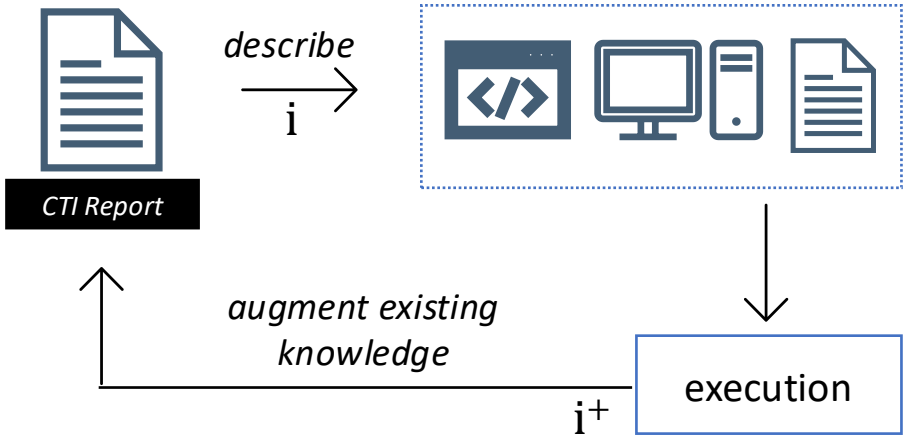
Privacy barriers prevent the validation of real-world data, while available public datasets remain outdated and incomplete

Cyber Threat Intelligence Reports

CTI reports lack the structure required to translate narratives into computable datasets

Insight: Augmenting Intelligence via Reproduction

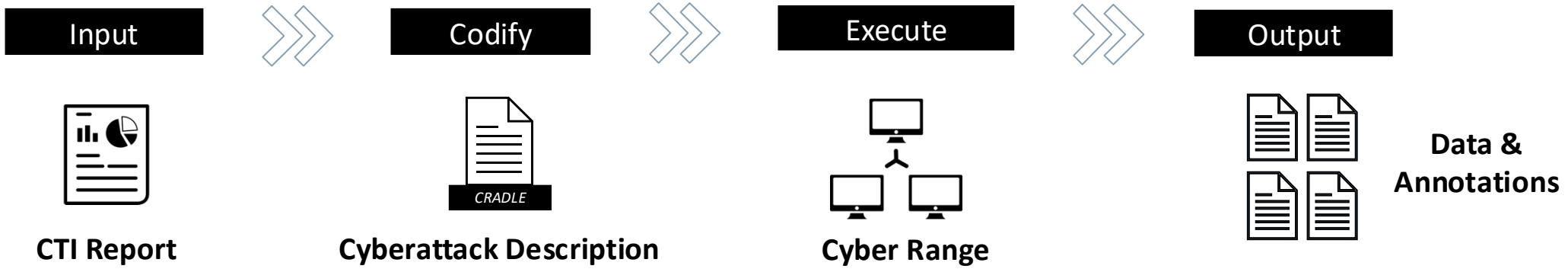
- Static indicators from Cyber Threat Intelligence (CTI) reports can be translated into executable specifications
- By treating them as code, we can transform sparse text into rich, empirical evidence
- Leverage known indicators to generate and augment missing information



Given a set of indicators i, can we generate attack datasets that contain more indicators i^+ where $i \in i^+$?

RECAP: End-to-end Dataset Generation Framework

- **RE**construction of **C**yberattack **A**ctivity & **P**rovenance (**RECAP**)
- **Key Idea:** Systematically instantiate cyberattack narratives into a controlled cyber range using CRADLE specification for dataset generation



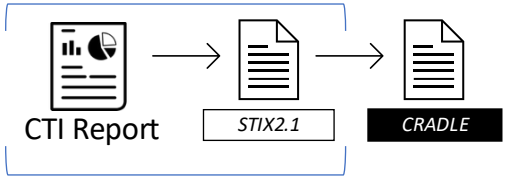
- Parse CTI Report
- Extract Primitives

- Map primitives to CRADLE specification
- Executable blueprint of the cyberattack

- Provision and deploy computing infrastructure
- Orchestrate attack

- Capture system logs, audit logs, network captures, memory dump
- Automated data labeling

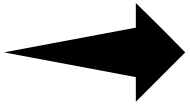
Capturing Cyberattack Semantics



- **Input Challenge**
  - CTI reports are written in natural language, making them ambiguous, heterogeneous, and non-executable
- **Semantic Extraction**
  - *Entity* Primitive – e.g., tools, malware, files
  - *Action* Primitive – e.g., execution behaviors

When run, it will automatically **load goopdate.dll** due to search order hijacking. **Goopdate.dll** is a highly obfuscated **loader** whose ultimate purpose is to load a **Cobalt Strike stager** into memory and then execute it. The Cobalt Strike stager will simply try to **download and execute a shellcode** from a **remote server**, in this case using the following URL …

APT32 CTI report from AttacKG [ESORICS'22]



```
{
    "type": "bundle",
    "id": "bundle--c3d...",
    "spec_version": "2.1",
    "objects": [
      {
        "type": "malware",
        "id": "malware--012...",
        "name": "Goopdate.dll Loader",
        "is_family": false,
        "description": "A highly obfuscated loader
designed to load a Cobalt Strike stager into memory.",
        "malware_types": ["loader"]
      },
      {
        "type": "tool",
        "id": "tool--23...",
        "name": "Cobalt Strike",
        "tool_types": ["exploitation"],
        "description": "Cobalt Strike stager used to
download and execute shellcode from a remote server."
      }, …
```

APT32 CTI report in STIX2.1

Capturing Cyberattack Semantics

Transforming static threat intelligence into a testbed-agnostic executable blueprint

```
{
    "type": "bundle",
    "id": "bundle--c3d...",
    "spec_version": "2.1",
    "objects": [
      {
        "type": "malware",
        "id": "malware--012...",
        "name": "Goopdate.dll Loader",
        "is_family": false,
        "description": "A highly obfuscated loader
designed to load a Cobalt Strike stager into memory.",
        "malware_types": ["loader"]
      },
      {
        "type": "tool",
        "id": "tool--23...",
        "name": "Cobalt Strike",
        "tool_types": ["exploitation"],
        "description": "Cobalt Strike stager used to
download and execute shellcode from a remote server."
      }, …
```

APT32 CTI report in STIX2.1

entities → *environment*

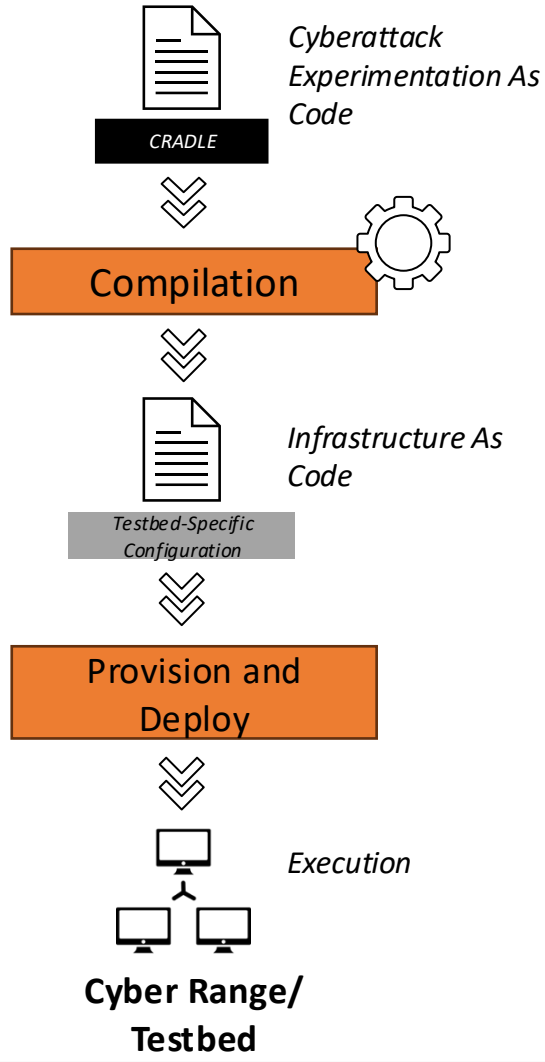
behaviors → *event*

```
instances() >
   instance("TargetUserPC"),
   instance("CommandControlServer").

instance("TargetUserPC") >
   os("windows", "10"),
   object("SpearPhishingEmail"),
   object("GoopdateDLLLoader"),
   object("CobaltStrike").

instance("CommandControlServer") >
   os("alpine", "3.19"),
   object("C2Scripts").
…
```

APT32 CRADLE specification

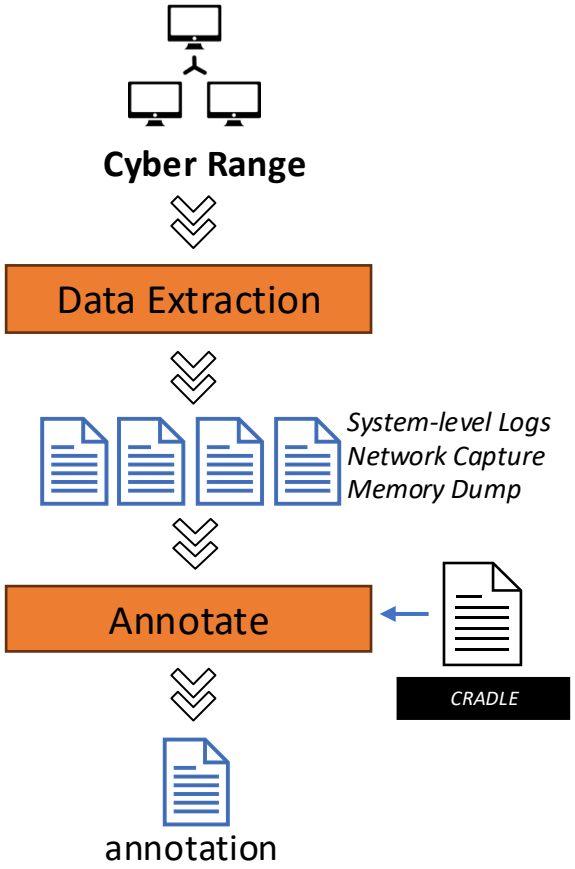
Instantiating CEaC: From Specification to Orchestration

- **Cyber Experimentation As Code**
  - Testbed-agnostic cyberattack definitions in CRADLE
- **Compiler**
  - Translate high-level constructs into low-level testbed-specific scripts
  - *Analogy* – compiling C++ into machine code for specific CPU
- **Provision and Deploy**
  - Provision the computing resources on the testbed
    - E.g., preparing the operating systems, networks, etc.
  - Deploy the computing resources for cyberattack execution
  - Orchestrate events throughout the cyber range



Dataset & Annotation

- **Comprehensive Artifact Collection**
  - **System-Level:** System logs, application logs, auth logs, audit logs
  - **Network-Level:** Full packet captures (PCAP)
  - **State-level:** Memory dumps capturing resident malware artifacts
- **Automated Labeling**
  - Leverages the event timeline in CRADLE specification as the ground truth to label the data
  - **Mechanism:** Correlates the orchestration timeline with artifact timestamps
  - **Annotation:** Precise, event-level labels linking specific log entries to specific attack behaviors



Evaluation

- **Experiment Setup**
  - **Reproduce five APT campaigns**
    - APT28, APT28-Variant, APT29, APT32, APT32-Variant
  - **Input Sources**
    - CTI reports from STIXNet [ARES'23] and AttacKG [ESORICS'22]
  - **Cyber Range**
    - Local testbed – Vagrant and Ansible
    - SPHERE testbed – Merge Experiment Model Language
  - **Instrumentation**
    - Windows: Sysmon, Event Viewer, Pktmon
    - Linux: auditd, sysdig, tcpdump
- **Evaluation Criteria**
  - ***Construction*** – How effective can RECAP generate heterogeneous datasets with granular annotations?
  - ***Efficiency*** – How efficient is the automated reproduction pipeline?
  - ***Indicators*** – Do the generated dataset contain valid indicators detectable by standard security ruleset?

Dataset Scale & Heterogeneity

Reproducing APT cyberattacks described from CTI report

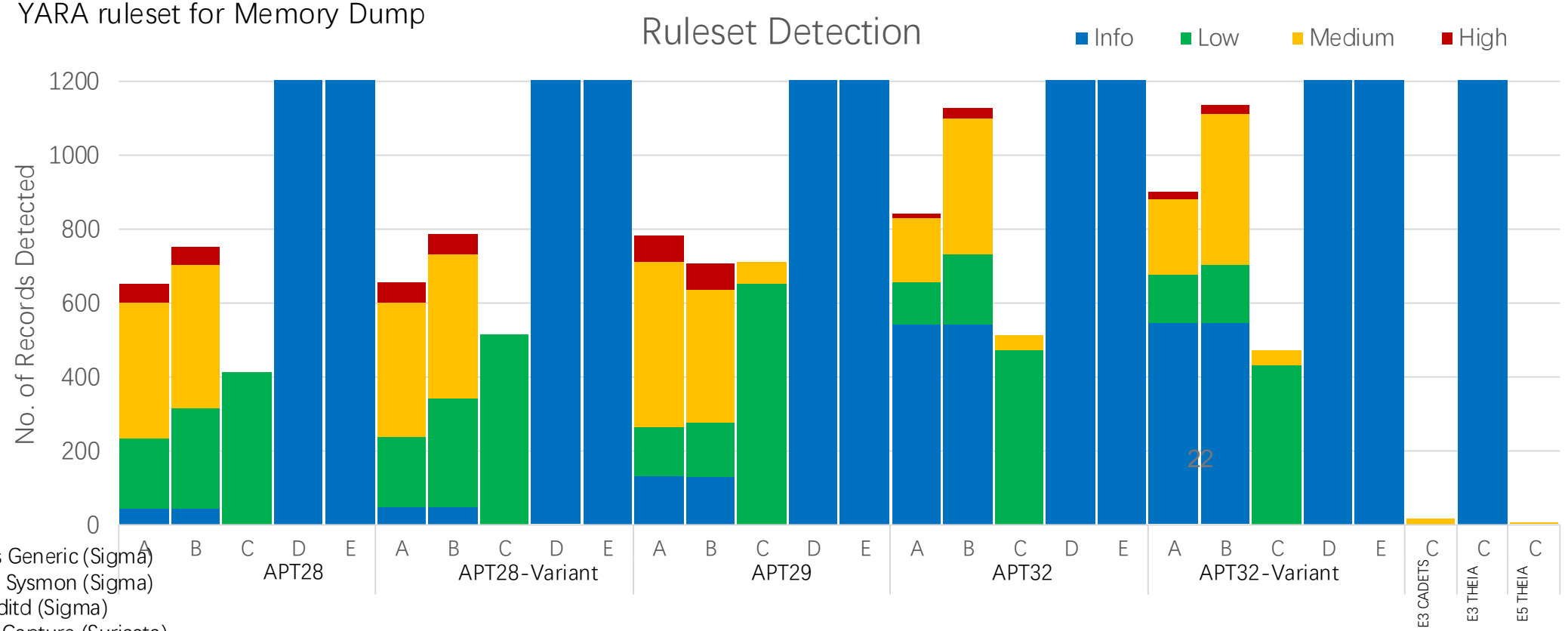
APT	System Logs				Provenance Graph	Network Capture	Memory Dump
	Auditd	Sysdig	Sysmon Event	Others			
APT28	Records: 227991 Annotated: 3885	Records: 1117727 Annotated: 2	Records: 7809 Annotated: 1	Records: 52374	Nodes: 2063 Edges: 3230	Records: 253305 File Size: 91.72 MB	File Size: 8363.61 MB
APT28-Variant	Records: 237017 Annotated: 3880	Records: 1101915 Annotated: 2	Records: 7788 Annotated: 5	Records: 52468	Nodes: 2105 Edges: 3322	Records: 87076 File Size: 37.33 MB	File Size: 8363.61 MB
APT29	Records: 301966 Annotated: 3890	Records: 646316 Annotated: 2	Records: 1377 Annotated: 1	Records: 31588	Nodes: 1612 Edges: 3764	Records: 5972 File Size: 2.47 MB	File Size: 10431.48 MB
APT32	Records: 175148 Annotated: 1968	Records: 471262 Annotated: 4	Records: 2356 Annotated: 3	Records: 40732	Nodes: 3520 Edges: 9014	Records: 123350 File Size: 56.66 MB	File Size: 8363.61 MB
APT32-Variant	Records: 175047 Annotated: 0	Records: 430395 Annotated: 0	Records: 4052 Annotated: 6	Records: 42291	Nodes: 4582 Edges: 9402	Records: 138962 File Size: 63.53 MB	File Size: 8363.61 MB

RECAP able to generate diverse datasets at scale with annotations based on heterogeneous cyberattack description

Data Validity and Quality

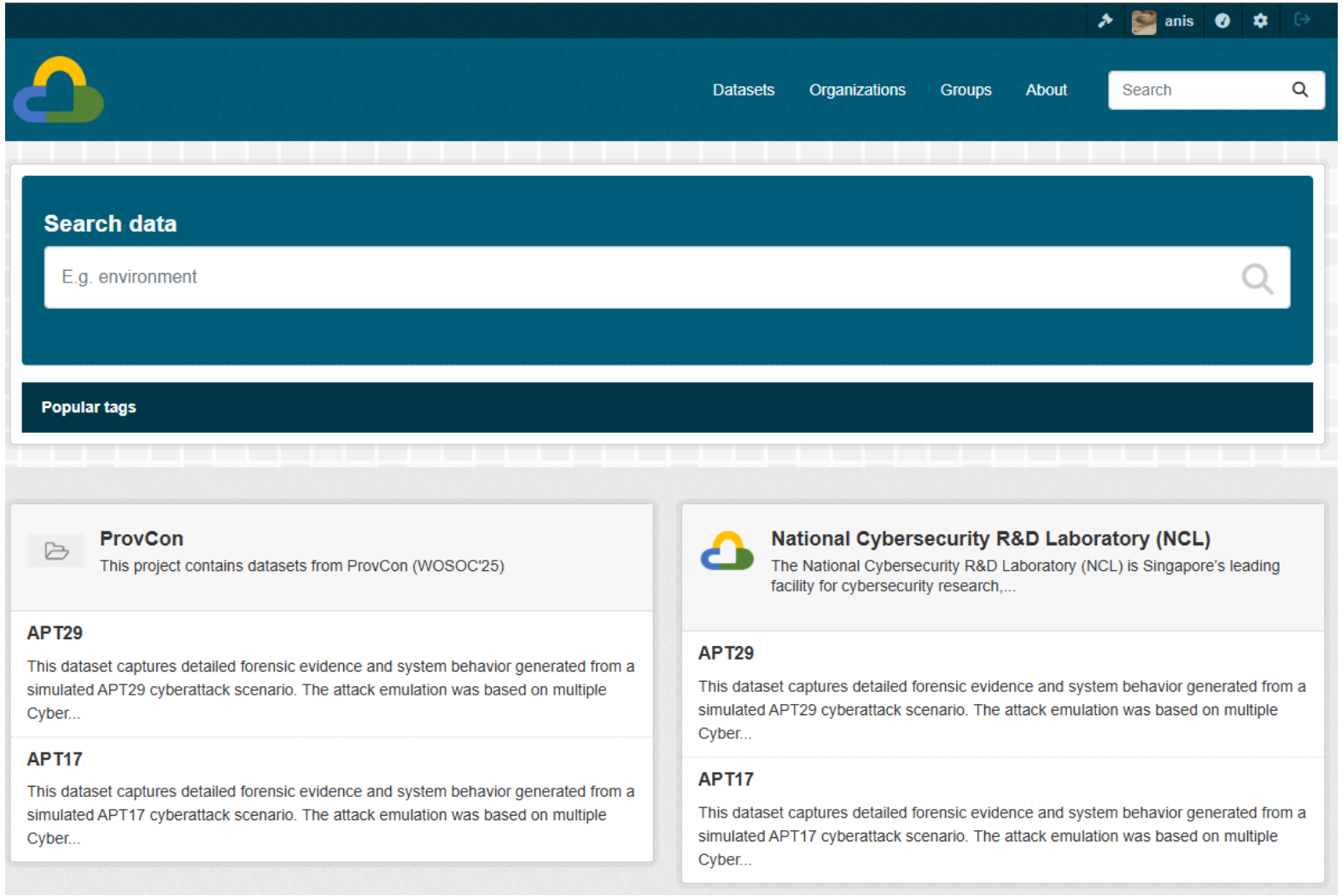
Validation using Standard Rulesets

- Sigma ruleset for System-level Logs
- Suricata ruleset for Network Capture
- YARA ruleset for Memory Dump



A = Windows Generic (Sigma)
B = Windows Sysmon (Sigma)
C = Linux Auditd (Sigma)
D = Network Capture (Suricata)
E = Memory Dump (YARA)

Dataset Collection at NCL

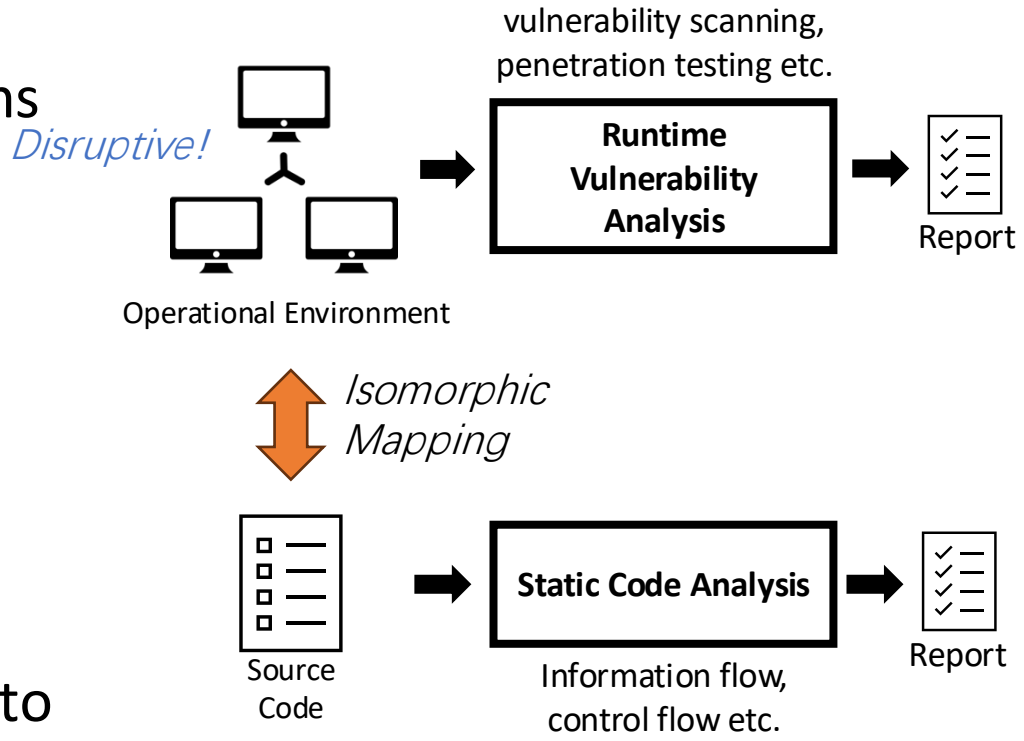


JELAS

Reasoning Over Infrastructure

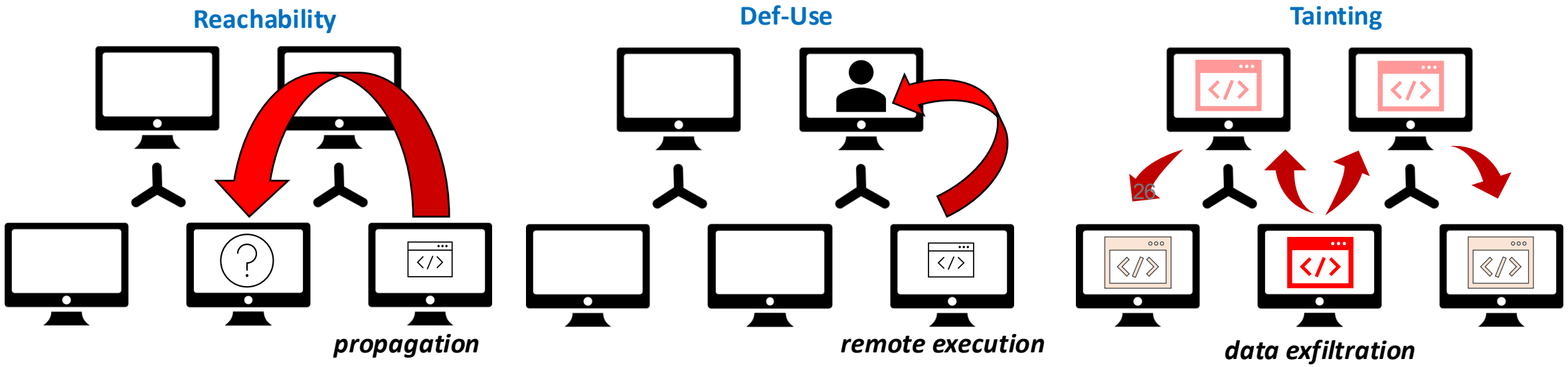
Insight: Infrastructure as A Debugable Program

- **Runtime Environment Analysis**
  - Requires active scanning of live systems
  - Limited to visible surfaces
  - Missing latent paths
- **The Static Analysis Paradigm**
  - Treating environment specification as source code
  - Enables the application of *Program Analysis* primitives (e.g., control flow) to discover infrastructure vulnerabilities



Mapping Program Analysis to Infrastructure

- Code-based analysis involves:
  - **Program Code → Infrastructure Specification**
  - **Variables and State → Computer Instance and Configurations**
  - **Control Flow → Network Connectivity**
  - **Data Flow → Artifact Propagation**

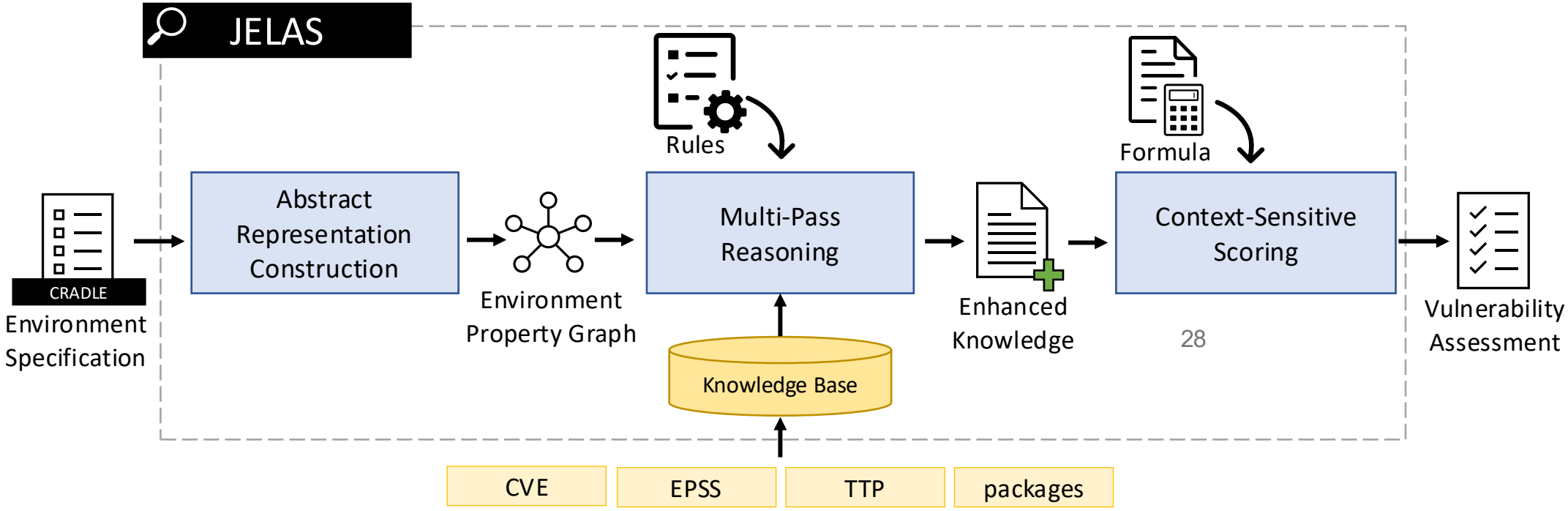


Analyzing Codified Infrastructure

- **Static Analyzers**
  - Detects security smells and syntax errors in Infrastructure as Code (IaC) scripts
    - Puppet [ICSE'19, ASE'22, MSR'16], Ansible [ASE'22, TOSEM'21], Chef [ASE'22, TOSEM'21]
  - **Gap:** security smells does not imply vulnerability
    - identify poor coding practices but lacks the network context to determine reachability
- **Semantic-based Analysis**
  - Models Infrastructure as Code (IaC) as graphs
    - Data Flow Graph [TACAS'21], Dependency Graph [SecDev'23]
  - **Gap:** focuses on software dependencies, missing multi-hop network paths required for lateral movement
- **Logic-based Approach**
  - Reason over facts about computing components to verify abstract system models
  - MulVAL [Security'05] and its extensions [2005-2021]
  - **Gap:** abstraction vs. fidelity trade-off
    - oversimplifies the attack primitive to make the analysis computationally feasible

JELAS: Reasoning Over Infrastructure

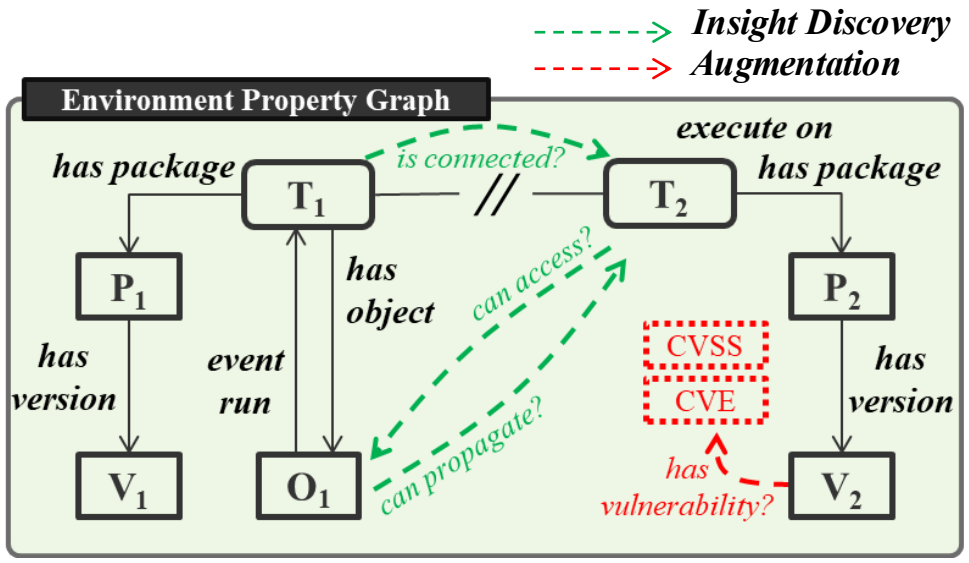
- **J**ustifiable **E**nvironment **L**evel **A**nalysis **S**ystem (**JELAS**)
- **Key Idea:** Environment-aware static analysis framework to discover insights based on CRADLE



Multi-Pass Reasoning

Enhance the current understanding about the computing infrastructure with related knowledge

- **Augmentation Pass**
  - Enrich the EPG with external knowledge (CVEs, EPSS, TTPs) to bridge the semantic gap
- **Insight Discovery**
  - Apply Datalog rules in Horn Clauses
  - Fixpoint Computation: The engine runs iterative cycles
    - Pass 1 finds direct links
    - Pass 2 finds transitive paths
    - The loop continues until the full graph is mapped and no new facts derived
  - Example: Applying **`CanConnectTo`** rule to find direct and transitive relationship between two instances



Transitive Reachability Rule

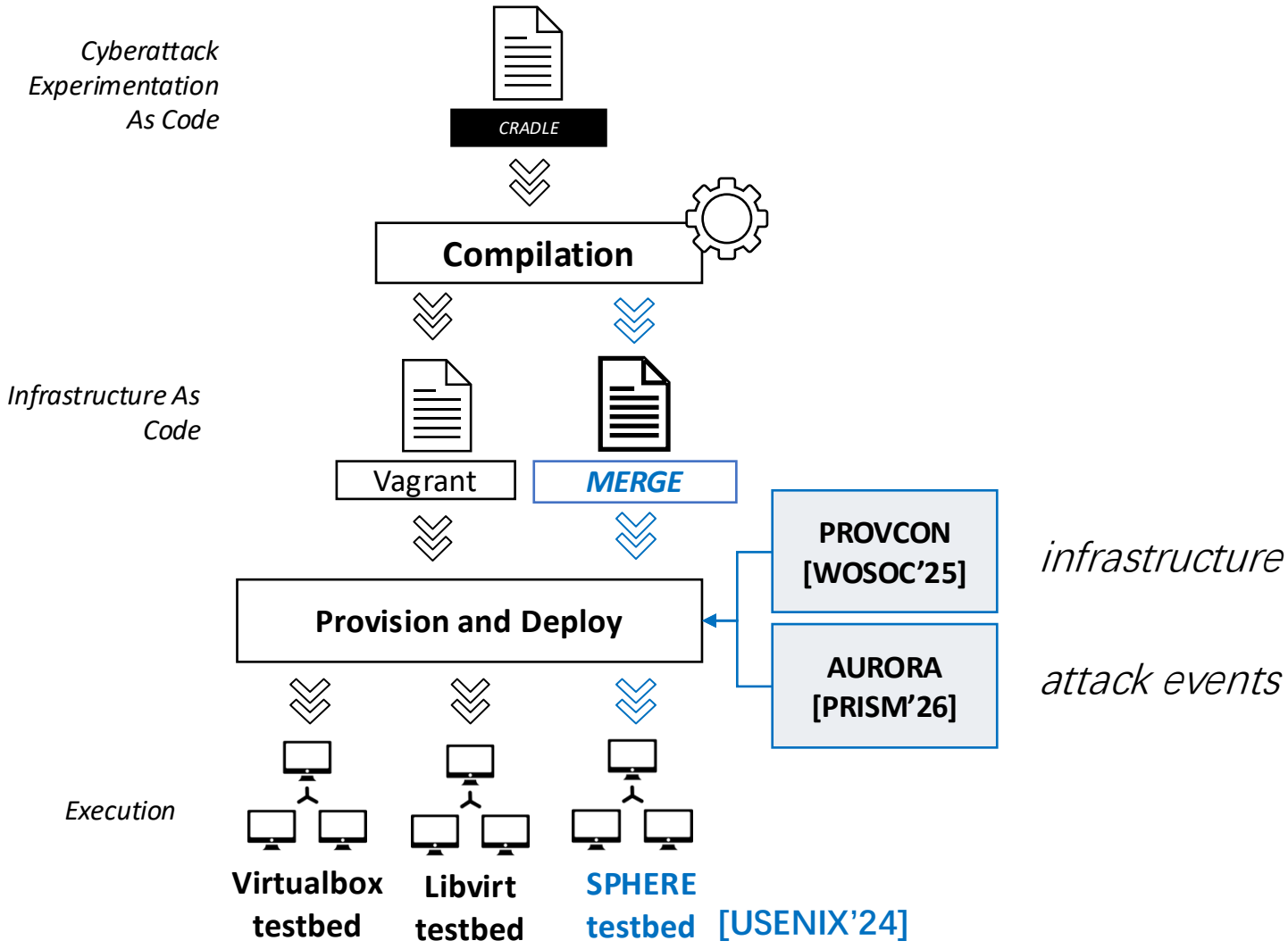
```
CanConnectTo(T1, T3, Transitive) :-
    CanConnectTo(T1,T2),
    CanConnectTo(T2,T3),
    T1 ≠ T2 ≠ T3.
```

CRADLE to SPHERE

Deploying on SPHERE

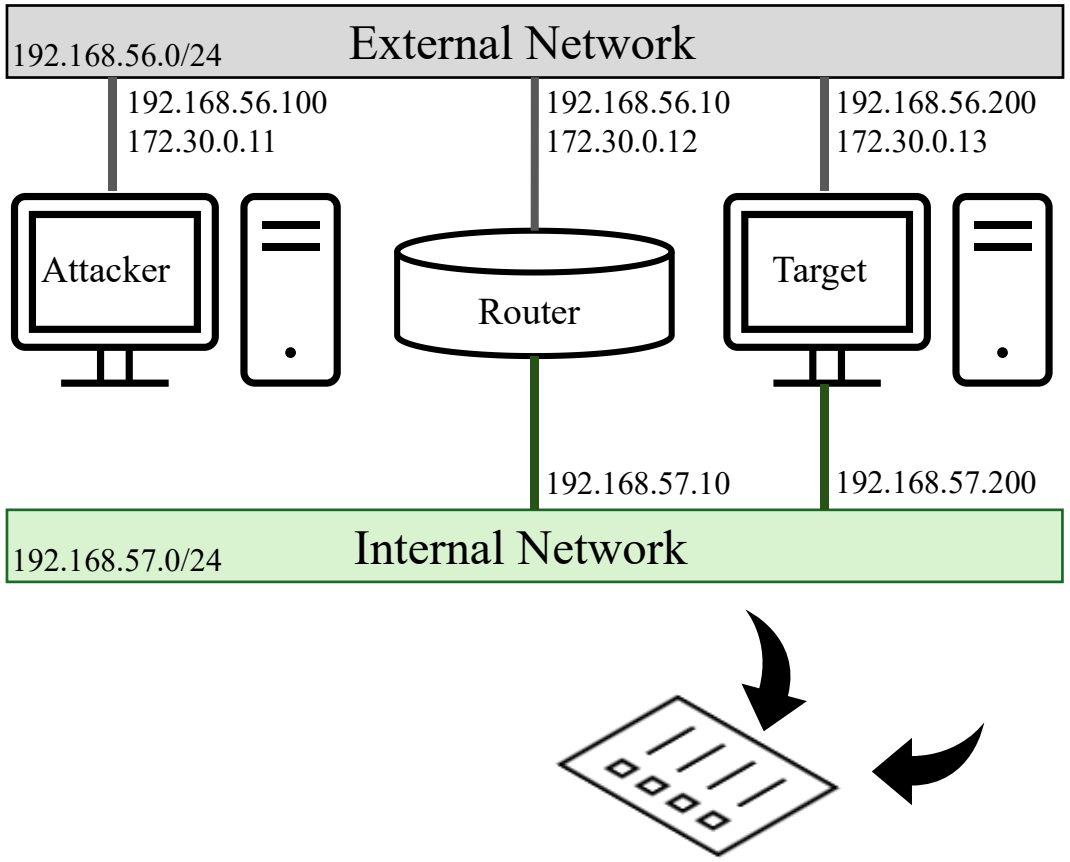
- CRADLE's testbed-agnostic capabilities by deploying on SPHERE testbed
- Integrated external research components to demonstrate modularity
  - RECAP & PROVCON [WOSOC'25] – for deployment
  - AURORA [Wang'24] - Generate attack chain with ***16 attack steps*** according to CTI report
- Compiles CRADLE specification to *MERGE Experiment Model Language*

Portability and Interoperability



Cyber Design

infrastructure



attack events

No.	Event Description
1	Attacker starts Python HTTP server
2	Target downloads payload from attacker
3	Target executes payload
4	Attacker execute whoami on target to verify current user
5	Attacker execute ping on target
6	Attacker execute ls on target to list files
7	Attacker execute ps on target to list processes
8	Attacker execute netstat on target based on selected filters
9	Attacker query environment variable using printenv
10	Attacker execute pwd to verify current directory
11	Attacker execute ifconfig to check network configuration
12	Attacker append persistence command to \$HOME/.shrc
13	Victim executes payload for privilege escalation
14	Attacker take a screenshot of the victim
15	Attacker execute mkdir -p to create a directory on the target
16	Attacker execute tar -cvzf to compress a folder to a file

Encode as Cyberattack Experimentation As Code

CRADLE to SPHERE Example

CTI Report – APT17 Deputy Dog

CTI Report

On February 11, FireEye identified a zero-day exploit (CVE-2014-0322) being served up from the U.S. Veterans of Foreign Wars' website (vfw[.]org). We believe the attack is a strategic Web compromise targeting American military personnel amid a paralyzing snowstorm at the U.S. Capitol in the days leading up to the Presidents Day holiday weekend. Based on infrastructure overlaps and tradecraft similarities, we believe the actors behind this campaign are associated with two previously identified campaigns (Operation DeputyDog and Operation Ephemeral Hydra).

This blog post examines the vulnerability and associated attacks, which we have dubbed "Operation SnowMan."

Exploit/Delivery analysis

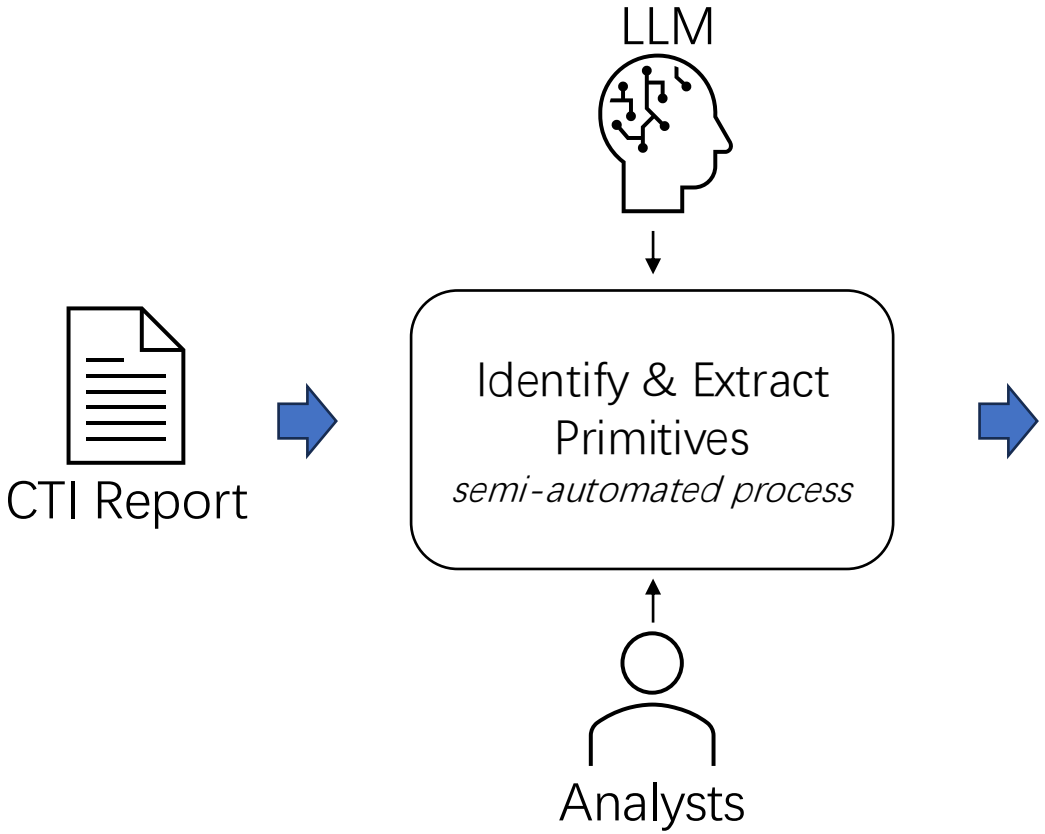
After compromising the VFW website, the attackers added an iframe into the beginning of the website's HTML code that loads the attacker's page in the background. The attacker's HTML/JavaScript page runs a Flash object, which orchestrates the remainder of the exploit. The exploit includes calling back to the IE 10 vulnerability trigger, which is embedded in the JavaScript. Specifically, visitors to the VFW website were silently redirected through an iframe to the exploit at www.[REDACTED].com/Data/img/img.html.

Mitigation …

AttacKG (ESORICS'22)

CRADLE Specification

- Identify and extract attack primitives
  - Infrastructure, objects, events




```
instances() >
    instance("router"),
    instance("target"),
    instance("attacker").

instance("router") >
    os("ubuntu", "20.04"),
    config("linux-router"),
    config("linux-tcpdump"),
    config("linux-auditd"),
    config("linux-sysdig"),
    config("linux-mail"),
    description("this is a router").

instance("attacker") >
    os("ubuntu", "20.04"),
    object("startweb"),
    object("webhost"),
    object("phish"),
    object("zxshell"),
    config("linux-python3"),
    config("python3-pip"),
    config("attacker"),
    config("linux-vsftpd"),
    config("linux-tcpdump"),
    config("linux-auditd"),
    config("linux-sysdig"),
    description("this is an attacker").

instance("target") >
    os("windows", "2019"),
    object("exploit"),
    object("sysinfo"),
    object("exfiltrate"),
    config("win-allow-ports"),
    config("win-pktmon"),
    config("win-audit"),
    config("win-sysmon").

networks() >
    network("InternalNetwork"),
    network("PublicFacingNetwork").

network("InternalNetwork") >
    subnet("192.168.56.0/24"),
    endpoint("router", "192.168.56.10"),
    endpoint("attacker", "192.168.56.100"),
    endpoint("target", "192.168.56.200").
```

APT17 CRADLE Specification

Integrating External Components

- AURORA (PRISM'26)
  - Generate artifacts to be used in CRADLE

Artifact that execute the attack activities described in the CTI report

https://dx.doi.org/10.14722/prism.2026.23021

CRADLE Update

- Update APT17 CRADLE specification with the `exploitpy` artifact generated by AURORA

- Include `exploitpy` and its corresponding exploit wrapper as part of the attack chain

```
16  instance("attacker") >
17      os("ubuntu", "20.04"),
18      object("exploitpy"),
19      object("exploit"),
20      config("sphere-linux-python3"),
21      config("sphere-linux-sliver-server"),
22      config("sphere-linux-auto_deploy"),
23      config("sphere-linux-zer0cool"),
24      config("sphere-deploy-config"),
25      config("sphere-modification-sliver"),
26      config("sphere-linux-auditd"),
27      config("sphere-linux-tcpdump").
28
29  instance("target") >
30      os("ubuntu", "20.04"),
31      config("sphere-linux-auditd"),
32      config("sphere-linux-tcpdump").
33
34  networks() >
35      network("ExternalNetwork"),
36      network("InternalNetwork").
37
38  network("ExternalNetwork") >
39      subnet("192.168.56.0/24"),
40      endpoint("router", "192.168.56.10"),
41      endpoint("attacker", "192.168.56.100"),
42      endpoint("target", "192.168.56.200").
43
44  network("InternalNetwork") >
45      subnet("192.168.57.0/24"),
46      endpoint("router", "192.168.57.10"),
47      endpoint("target", "192.168.57.200").
48
49  events() >
50      preEvent(),
51      mainEvent(),
52      postEvent().
53
54  mainEvent() >
55      event("1").
56
57  event("1") >
58      instance("attacker"),
59      needRoot("true"),
60      subject("bash", ""),
61      runObject("exploit", ""),
62      pauseBeforeRun("0"),
63      pauseAfterRun("5"),
64      waitfor("false"),
65      scheduleExecution("2024-06-19T08:30:00Z"),
66      description("Run attack chain").
67
68  object("exploitpy") >
69      location("${uriRemote}/APT17-sphere/placeholder/attack_chain.py").
70
71  object("exploit") >
72      location("${uriRemote}/APT17-sphere/placeholder/run-exploit.sh").
```

artifact generated by AURORA

```
cdl@swl:~/artifact/APT17-sphere$ cat run-exploit.sh
python attack_chain.pycdl@swl:~/artifact/APT17-sphere$ cat attack_chain.py

import asyncio
from rich.console import Console
from rich.prompt import Confirm
from rich.panel import Panel
from typing import Dict
console = Console()
user_params: Dict[str, str] = {}
def print_welcome_message():
    console.print(
        Panel(
            "[bold blink yellow]?? Welcome to Attack Execution Wizard[/]",
            title="[bold green]Hello[/]",
            subtitle="[bold blue]Let's Begin[/]",
            expand=False,
        )
    )
def print_finished_message(message="Command completed!??", status="info"):
    console.print(f"[bold green][FINISHED][/bold green] {message}")
def confirm_action(prompt: str = "Keep going with the next attack step?") -> bool:
    styled_prompt = f"[bold bright_cyan]{prompt}[/]"
    return Confirm.ask(
        styled_prompt,
        default="y",
        choices=["y", "n"],
        show_default=False,
    )
async def main():
```

Compiling to Local Testbed

compilation

CRADLE → Intermediate Representation → Vagrant → Libvirt/VirtualBox (local)

```
(venv) cdl@swl:~/cradle-main$ head -5 files/input/APT17.cradle
metadata() >
    name("APT17"),
    eventType("sequence"),
    repositoryRemote("https://example.com/repository"),
    object("sysinfo"),
(venv) cdl@swl:~/cradle-main$ python3 compiler/cradle-parser.py \
    -g \
    -i files/input/APT17.cradle \
    -o files/output/APT17.yml
/home/cdl/cradle-main/compiler/cradle-parser.py:60: SyntaxWarning: invalid escape sequence '\)'
  rhsDict["name"] = re.sub('\)\)$', ')', rhsDict["name"])
/home/cdl/cradle-main/compiler/cradle-parser.py:62: SyntaxWarning: invalid escape sequence '\.'
  rhsDict["name"] = re.sub('\.\)$', '', rhsDict["name"])
(venv) cdl@swl:~/cradle-main$ head -5 files/output/APT17.yml
events:
  mainEvent:
  - delay_after: '10'
    delay_before: '0'
    description: Sending phishing email to deploy initial exploit vector
(venv) cdl@swl:~/cradle-main$
```

Compiling to Local Testbed

assembler

CRADLE → **Intermediate Representation → Vagrant → Libvirt/VirtualBox (local)**

```
(venv) cdl@swl:~/cradle-main$ python3 assembler/bin/run.py \
    --cradle_file files/output/APT17.yml \
    --screenplayName APT17 \
    --testbed local
(venv) cdl@swl:~/cradle-main$ cd assembler/bin/output/APT17/Deployment_For_local/APT17-experiment/localhost &&
ll
total 92
drwxr-xr-x 6 cdl cdl  4096 Jul 24 22:52 ./
drwxr-xr-x 3 cdl cdl  4096 Jul  9 17:29 ../
drwxr-xr-x 2 cdl cdl  4096 Jul  9 17:29 attacker/
-rwxr-xr-x 1 cdl cdl  5008 Jul 24 22:53 bm_script*
-rw-r--r-- 1 cdl cdl  2859 Jul 24 22:53 bootstrap.sh
-rw-r--r-- 1 cdl cdl  2687 Jul 24 22:53 event_playbook.yml
-rwxr-xr-x 1 cdl cdl  2386 Jul 24 22:53 fix_windows_hosts.sh*
-rw-r--r-- 1 cdl cdl  1117 Jul 24 22:53 hosts
-rw-r--r-- 1 cdl cdl 24496 Jul 24 22:53 provision_playbook.yml
drwxr-xr-x 2 cdl cdl  4096 Jul  9 17:29 router/
drwxr-xr-x 2 cdl cdl  4096 Jul  9 17:29 target/
-rw-r--r-- 1 cdl cdl  1810 Jul  9 17:29 truncate.yml
-rw-r--r-- 1 cdl cdl   617 Jul 24 22:53 update.sh
drwxr-xr-x 5 cdl cdl  4096 Jul  9 17:29 .vagrant/
-rw-r--r-- 1 cdl cdl  4537 Jul 24 22:53 Vagrantfile
-rw-r--r-- 1 cdl cdl   603 Jul 24 22:53 value_assignor.yml
(venv) cdl@swl:~/cradle-main/assembler/bin/output/APT17/Deployment_For_local/APT17-experiment/localhost$
```

Compiling to MERGE for SPHERE testbed

compilation

CRADLE → Intermediate Representation → MERGE

```
(venv) cdl@swl:~/cradle-main$ head -5 files/input/APT17sphere.cradle
metadata() >
    name("APT17sphere"),
    eventType("sequence"),
    repositoryRemote("https://example.com/repository"),
    object("sysinfo"),
(venv) cdl@swl:~/cradle-main$ python3 compiler/cradle-parser.py \
    -g \
    -i files/input/APT17sphere.cradle \
    -o files/output/APT17sphere.yml
/home/cdl/cradle-main/compiler/cradle-parser.py:60: SyntaxWarning: invalid escape sequence '\)'
  rhsDict["name"] = re.sub('\)\)$', ')', rhsDict["name"])
/home/cdl/cradle-main/compiler/cradle-parser.py:62: SyntaxWarning: invalid escape sequence '\.'
  rhsDict["name"] = re.sub('\.\)$', '', rhsDict["name"])
(venv) cdl@swl:~/cradle-main$ head -5 files/output/APT17sphere.yml
events:
  mainEvent:
  - delay_after: '10'
    delay_before: '0'
    description: Sending phishing email to deploy initial exploit vector
(venv) cdl@swl:~/cradle-main$
```

Compiling to MERGE for SPHERE testbed

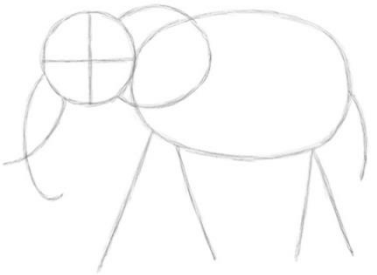
assembler

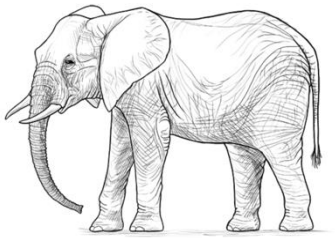
CRADLE → **Intermediate Representation → MERGE**

```
(venv) cdl@swl:~/cradle-main$ python3 assembler/bin/run.py \
    --cradle_file files/output/APT17-sphere.yml \
    --screenplayName APT17-sphere \
    --testbed sphere
(venv) cdl@swl:~/cradle-main$ cd assembler/bin/output/APT17sphere/Deployment_For_sphere/APT17sphere-experiment/
localhost && ll
total 128
drwxr-xr-x 2 cdl cdl  4096 May 20 16:03 ./
drwxr-xr-x 3 cdl cdl  4096 Jan 29 15:53 ../
-rwxr-xr-x 1 cdl cdl 30609 Mar 13 18:23 bootstrap.sh*
-rw-r--r-- 1 cdl cdl  1723 Mar 13 18:04 event_playbook.yml
-rw-r--r-- 1 cdl cdl 14492 Mar 13 18:04 extraction.yml
-rw-r--r-- 1 cdl cdl  1202 Mar 13 18:04 extract_logs-old.ps1
-rw-r--r-- 1 cdl cdl  1251 Mar 13 18:04 extract_logs.ps1
-rwxr-xr-x 1 cdl cdl   999 Mar 13 18:04 extract_logs.sh*
-rw-r--r-- 1 cdl cdl   399 Mar 13 18:20 hosts
-rw-r--r-- 1 cdl cdl    56 Mar 13 18:20 hosts.sphere
-rw-r--r-- 1 cdl cdl   496 Mar 13 18:04 model.py
-rw-r--r-- 1 cdl cdl  1418 Feb 14 14:10 network_config.yml
-rw-r--r-- 1 cdl cdl 24837 Mar 13 18:04 provision_playbook.yml
-rwxr-xr-x 1 cdl cdl  2687 Feb  3 10:38 test_network.sh*
-rw-r--r-- 1 cdl cdl  1810 Mar 13 18:04 truncate.yml
-rw-r--r-- 1 cdl cdl   563 Mar 13 18:04 value_assignor.yml
(venv) cdl@swl:~/cradle-main/assembler/bin/output/APT17sphere/Deployment_For_sphere/APT17sphere-experiment/loca
lhost$
```

```
(venv) cdl@swl:~/cradle-main/assembler/bin/output/APT17sphere/Deployment_For_sphere/APT17sphere-experiment/loca
lhost$ cat model.py
"""
Mergexp model for APT17sphere
Generated by CRADLE for Sphere testbed deployment
"""

from mergexp import *

# Create network topology
topo = Network("APT17sphere")

# Define nodes with image constraints
router = topo.node("router", image == "2004")
target = topo.node("target", image == "2004")
attacker = topo.node("attacker", image == "2004")

# Define connections between nodes
topo.connect([router, target, attacker])
topo.connect([router, target])

# Register experiment
(venv) cdl@swl:~/cradle-main/assembler/bin/output/APT17sphere/Deployment_For_sphere/APT17sphere-experiment/loca
lhost$
```

Conclusion





- CRADLE: A Language designed for Cyber Event Reproduction
  - RECAP: End-to-end Dataset Generation Framework
  - JELAS: System-wide analysis
  - CRADLE to SPHERE compilation
- As a high-level language, CRADLE aims to unify diverse sources of cyber experiment information and be “compiled” to drive different testbed technologies.

CURISITY

Understanding systems 理解系统
Abstracting knowledge 提炼知识
Connecting facts 参悟规律