DEW: Distributed Experiment Workflows



Jelena Mirkovic, **Genevieve Bartlett** and Jim Blythe
{mirkovic, bartlett, blythe}@isi.edu
USC Information Sciences Institute

Testbed Evaluation

- Vital for testing security solutions
- Testbed evaluation requires structured, rigorous and robust hypothesis testing
- Peer review: Communicate what/how

But, Sometimes More Art Than Science

Noble Goals in Testbed Experimentation

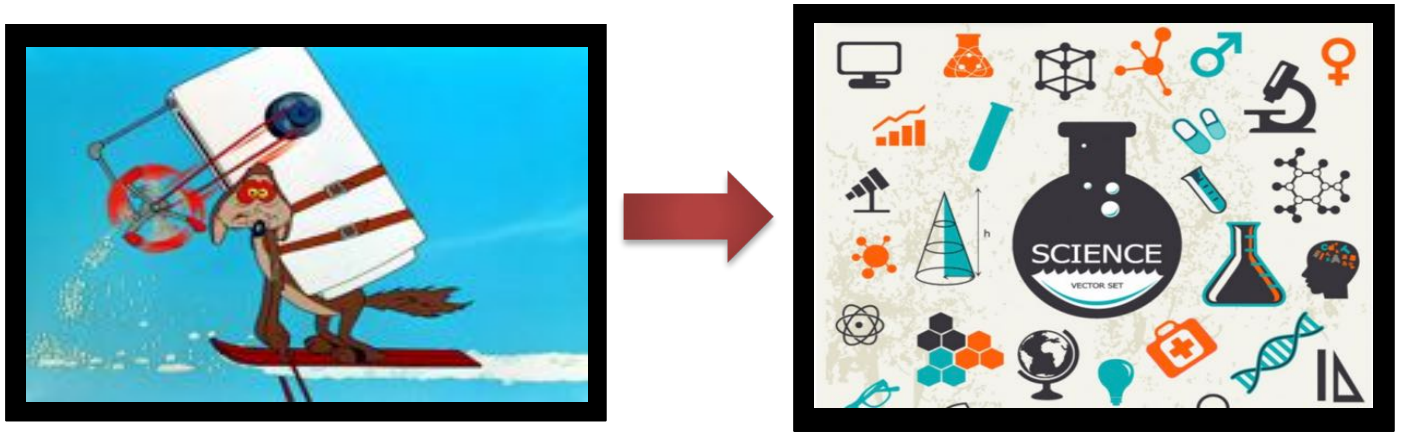
Less:

- Tedious + Manual
- Error prone

More:

- Automation
- Proactive error detection

Noble Goals in Testbed Experimentation



- Better artifact and documentation creation
- Repeatability and Reuse (needs portability)
- Proactively identify and address errors

Why are we not there yet?

Why are we not there yet?

- Experiment representations are lacking
  - Currently focus on topological structure and resources
  - Need standardized way to encode *behavior*

Why are we not there yet?

- Experiment representations are lacking
  - Currently focus on topological structure and resources
  - Need standardized way to encode *behavior*
- DEW: A way to represent experiments

Overview

- Distributed Experiment Workflows: DEW
- Automation through DEW
- Building with and on prior work
- UI/Demo

Overview

- Distributed Experiment Workflows: DEW
- Automation through DEW
- Building with and on prior work
- UI/Demo

Distributed Experiment Workflows

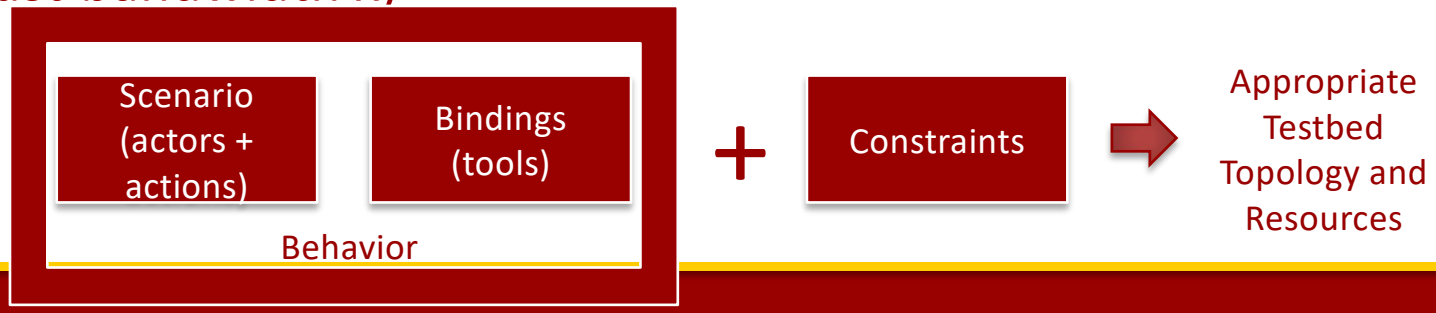
- Captures Full Experiment Description by drawing out only what matters
- behavior + resources/topology = experiment
- Strong separation between the behavior, the tools that enact that behavior and the topology the behavior is enacted on

Full Experiment Description

- **Works like a playscript:**
  - **Scenario**: The “What and who” (actions in an experiment, and the actors involved)
  - **Bindings**: The “How” (the tools, orchestration and configurations needed to carry out the what)
  - **Constraints**: The “Where” (such as on hardware x, os y, linked with *at least* bandwidth x)

Full Experiment Description

- **Works like a playscript:**
  - **Scenario**: The "What and who" (actions in an experiment, and the actors involved)
  - **Bindings**: The "How" (the tools, orchestration and configurations needed to carry out the what)
  - **Constraints**: The "Where" (such as on hardware x, os y, linked with *at least* bandwidth x)



Gist of a DEW Statement

- General: <Trigger(s)> <Actor(s)> <action(s)><signals>
- Examples:
  - Attacker startAttack
  - WHEN startWebserver WAIT t0 Attacker startAttack EMIT attackStarted
- Note: Actors != individual resources
  - E.g. An “attacker” role may be spread across multiple physical nodes
  - E.g. Multiple nodes acting in the same “client” role

DEW Goals

- High-level representation
- Generic language
- Self-contained representation
- Decouple behavior from topology and resources
- Structured representation

High-level Representation

- Human-readable (no, really…)
- Quick glance should tell you what the experiment does
- Enables humans to sort out what is interesting, useful and reusable

Generic Language

- Support a diverse range of experiments
- Focus now on cybersecurity and human modeling, but goal is to be broadly applicable

Self-contained Representation + Decouple Behavior

- Capture enough details to support automatic generation of experiment pieces for a range of testbeds
- Decouple topological structure and resources enables easy scaling and portability

Structured Representation

- Focus on the high-level first
  - Match natural flow for humans in understanding or describing a process
- Focus on only the important details
  - Constraints emphasize the most salient details in reconstructing the underlying resources for an experiment

Overview

- Distributed Experiment Workflows: DEW
- Automation through DEW
- Building with and on prior work
- UI/Demo

Automation

- **Generate -**

  DEW -> experiment

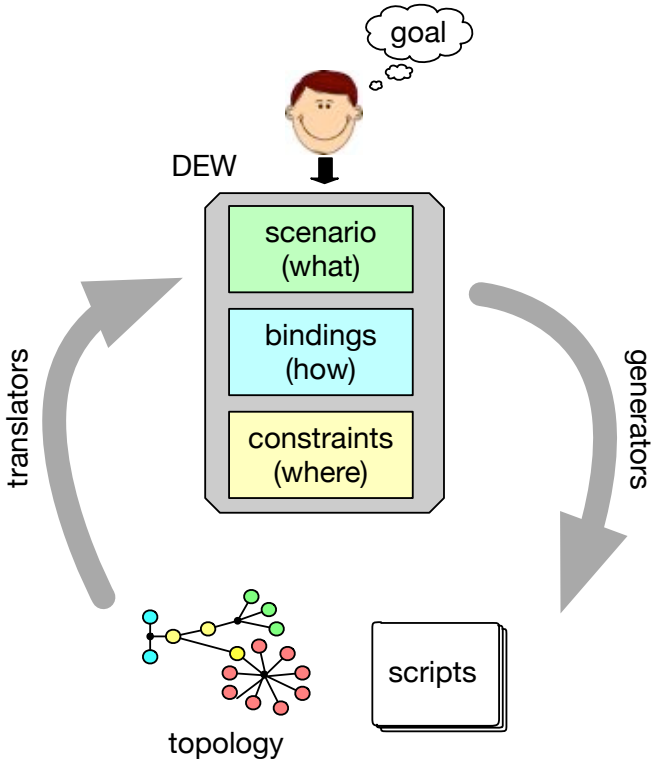
  – Scripted tools (including orchestration tools)

  – Topology descriptions

- **Translate –**

  experiment -> DEW

  (reverse process)



Generators

- May not produce fully featured scripts, but:
  - Provide structure for common variables for configuring and tuning
  - Structure for varying independent variables and producing runs of results
  - Offers point to decouple orchestration from other experiment tooling, enabling different orchestration to be inserted for different environments

Translators

- Work with how users work currently
- Benefit: potential eventual adoption, but if not, helps the experiment be sharable/portable
- Challenge: capture manual input in a meaningful way
  - Identify and prune paths of unproductive/undone input
  - Identify and capture varying independent variables

Overview

- Distributed Experiment Workflows: DEW
- Automation through DEW
- Building with and on prior work
- UI/Demo

Standing on the Shoulders

- Let's not insist on "stepping on the toes of those who came before us instead of climbing on their shoulders" – Dan Ingalls

Standing on the Shoulders

- Many inspiring works:
  - NS-based Experimentation Workbench (Eide et al.)
  - GPLMT (XML-based)
  - Grid computing workflows
- DEW
  - Higher-level language (much shorter descriptions)
  - Stronger abstraction from topology/resources
  - Translators/Generators enable building with and on other workflow tools

Overview

- Distributed Experiment Workflows: DEW
- Automation through DEW
- Building with and on prior work
- UI/Demo

Prototype UI: Key Features

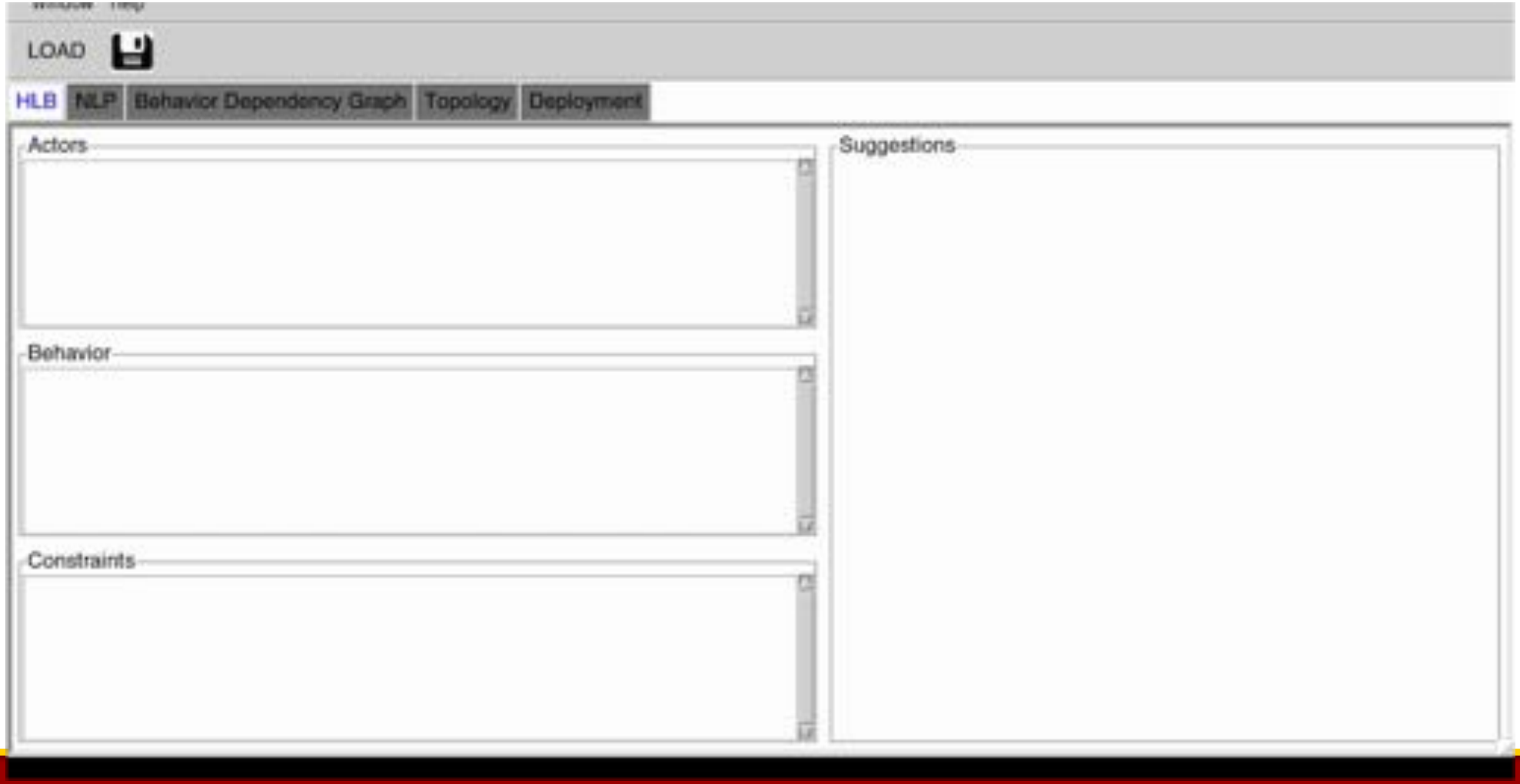
- Assisted text UI
  - Suggestions to help with DEW syntax
- Natural Language Processing
  - NLP->DEW
  - Challenging, but a first stab at living the dream
- DAG-based representation of event dependencies
- Topology depiction based on constraints
  - past experiences with DETER indicate users under-constrain, DEW fills in some guesses

Quick Demo: Set up

- Test of DoS defense deployed at a firewall
- Actors: webserver, firewall, attacker
- After the webserver is up and serving content, the attacker will begin an attack. Then the firewall will deploy defenses.
- In DEW:

```
Webserver startWebserver EMIT startWebserverSig
WHEN startWebserverSig Attacker startAttack EMIT startAttackSig
WHEN startAttackSig Firewall startDefences EMIT startDefencesSig
```

+ tool bindings + some constraints


LOAD
HLB
NLP
Behavior Dependency Graph
Topology
Deployment
Actors
Behavior
Constraints
Suggestions

Call to Action

- Help us develop DEW
  - Can you describe your experiment in DEW?
  - What's missing in DEW? What worked?
  - UI can help you play with the language
- Thanks:
  - Jelena Mirkovic, Genevieve Bartlett, Jim Blythe
    {mirkovic, bartlett, blythe}@isi.edu
  - Github: https://github.com/gbartlet/DEW