CTF: State-of-the-Art and Building the Next Generation

ASE 2017

Clark Taylor
Lawrence Livermore National Laboratory
University of Arizona

August 15, 2017

LLNL-PRES-737334
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

The University of Arizona

Lawrence Livermore National Laboratory

Motivation

- Cyber Defenders Program
- 8 years of Cyber Defenders CTF
  - Different competitions from Sandia and LANL
  - On-site and remote setups
- 3 years of Cybercraft
  - Summer long for Cyber Defenders
  - Events at library, National Science Bowl
  - Based on PicoCTF framework
- Results?
  - Surveys indicate generally good results

General Competition Goals

- Ease of use
  - Alternative target audiences
- Keep costs down
  - Hardware
  - Administration
- Competition realism
  - Policy
- Variety of modes
  - Engaging for a range of skill levels
- Research/evaluation outcomes
- Framework extensibility

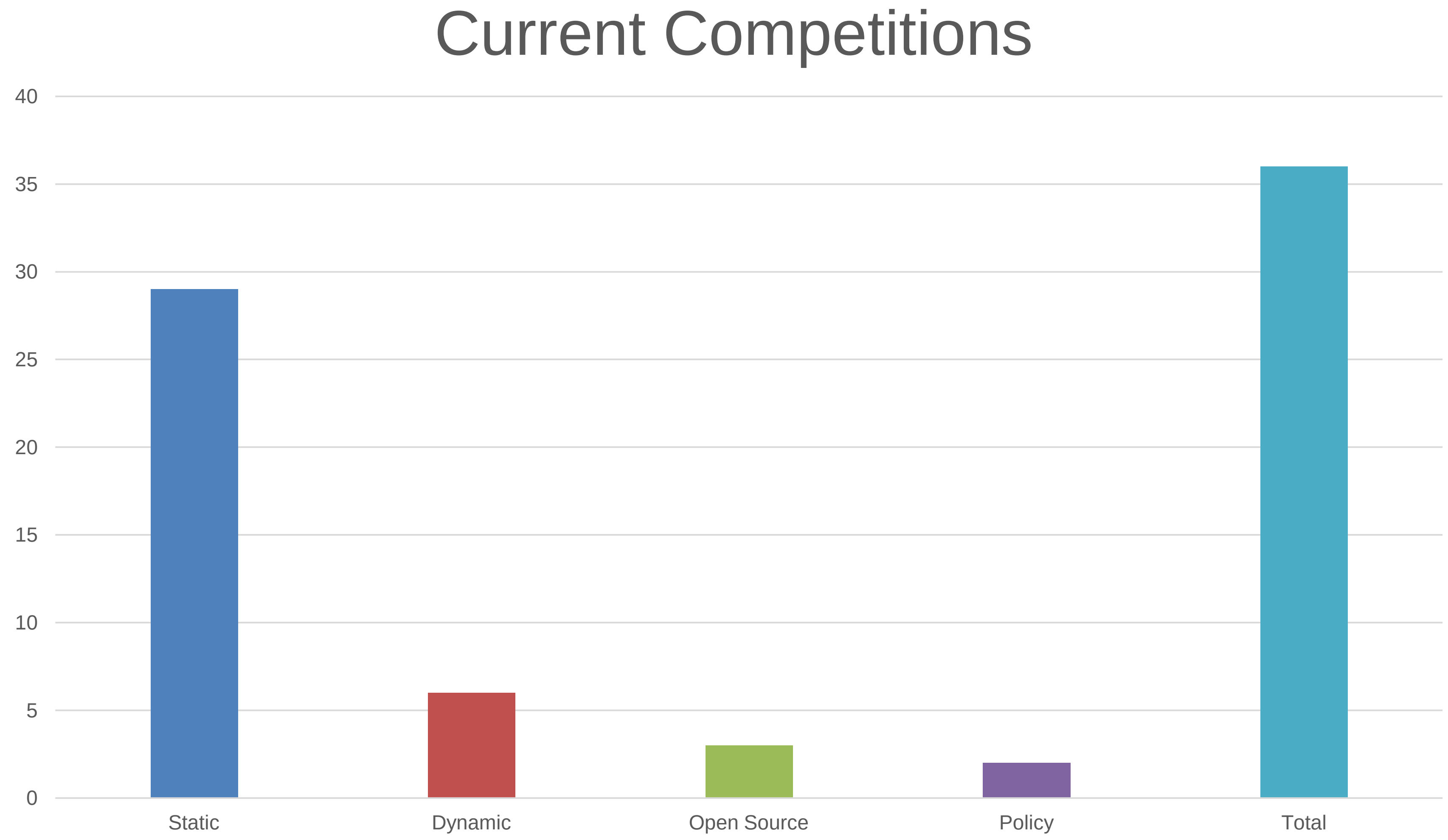
General Strategies for Cyber Defenders

- Hardware costs
  - Raspberry PI and Kali Linux
- Add storylines and custom content for realism
- Design content for training (Cybercraft)
- Collect surveys and summary data

Can we do Better?

- Cyber Defender competitions had shortcomings
- What other CTFs are out there?
- Studied 39 different CTFs
- Found commonalities
  - Framework vs monolithic
    - Open source?
  - Dynamic vs static challenges
    - “Challenges” refers to individual puzzles or tasks for which points are awarded
  - Policy topics

State of the Art

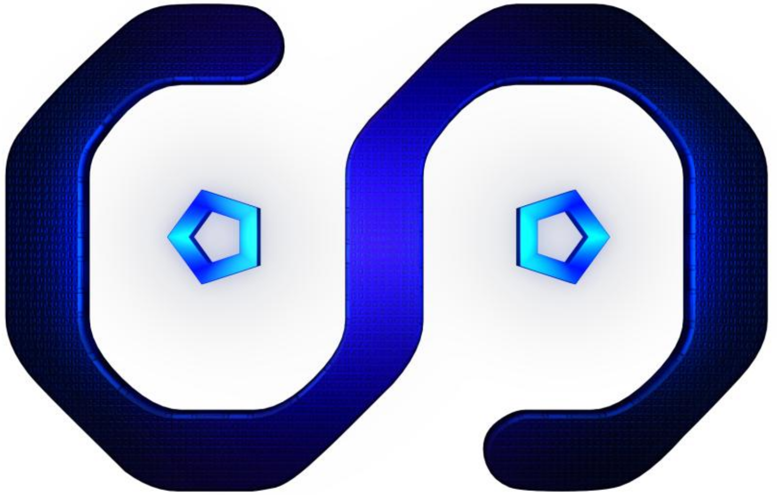


Shortcomings

- Ease of use
- Realism
  - Challenge types
  - Policy integration
- Training oriented modes
- Data collection
- Content development ease

Proposed Framework: Catalyst

- Introducing the Catalyst Security Challenge (CSC)
- Aims to solve these problems
- Currently in progress
  - Building component-by-component



Ease of Use

- Entirely web based
  - Hosted on a LAN, with network management (including VPN) on game server
  - Plaintext-and-buttons configurable
- Automated provisioning
  - Provision the game server from the internet
  - Provision other components from the game server
    - Includes provisioning for participant terminals
    - Components can be virtual
    - Components can be inexpensive hardware
- GUI managed
- Target audience: Lowest common denominator

Realism

- Support for static and dynamic content
  - HTTP-based extensible grading system API
- Highly configurable
  - Plaintext configuration includes challenge text and parameter configuration
- Policy challenges
  - Built-in support for policy-oriented types of challenges
  - Additional realism and configurability better enables policy challenges

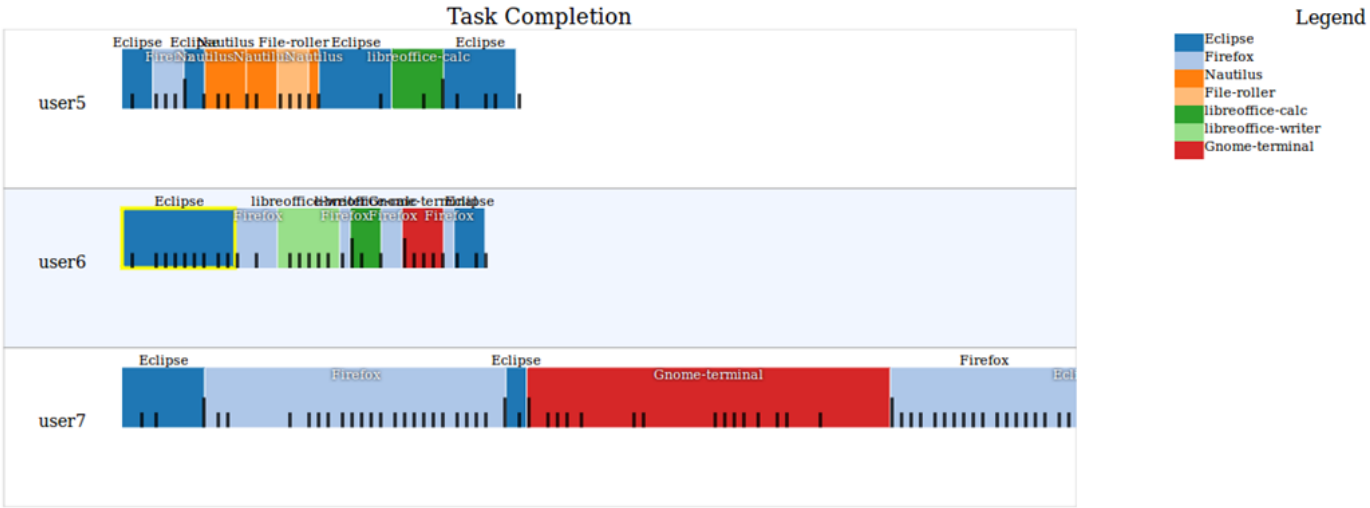
Training Modes

- Administrators can choose desired mode of competition
- Training modes disable features such as public scoreboards
- Support for challenge hints
- Challenge components may behave differently
  - — Content developers can access the current mode for their components via the grading API

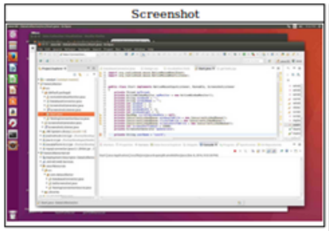
Data Collection

- Keylogger-on-steroids approach
  - Monitors endpoints
  - Installed automatically via provisioning
- Data curated on game server
- Visualization and filtering
- Goals:
  - Find novel approaches to solving challenges
  - Determine best practices and strategies
  - Evaluate efficacy of CTF for training/education
  - Evaluate participants

Data Collection: Current Visualization



Screenshot

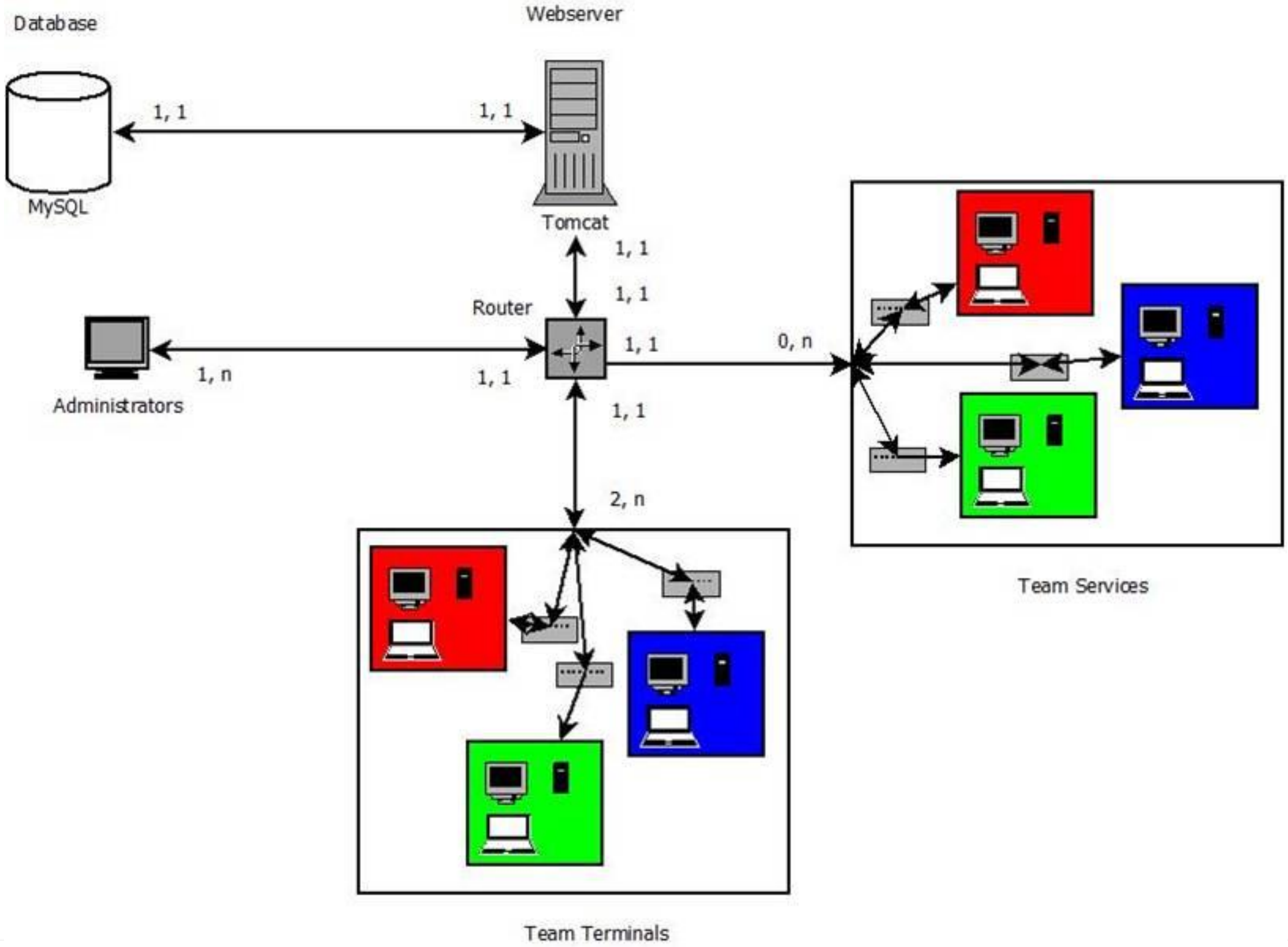


Details			
End Time	2016-12-08 21:52:38.0	End Time MS	11000
Input Location X	809	Input Location Y	809
Input Time	2016-12-08 21:52:28.0	Input Time MS	1000
Input Type	down	Memory Use	14
Number	0	PID	6966
RSS	1852160	Start	Dec07
Start Time	2016-12-08 21:52:27.0	Start Time MS	0
Stat	Sl+	TTY	pts/3
Time	33:26	User	root
Username	user6	VSZ	8388607
Window Class 1	Eclipse	Window Class 2	Eclipse
Window Height	846	Window Location X	159
Window Location Y	122	Window Name	Java EE - DataCollector/src/Start.java
Window Width	1424	XID	0x3400513

Content Development

- No particular design patterns
  - Just has to be compatible with host OS and support HTTP
- Game server provisioning launches software via OS
- All communication done via grading API

Architecture



Questions?

Lawrence Livermore National Laboratory