The Future for Security

Greg Morrisett
Jack & Rilla Neafsey Dean & Vice Provost
Cornell Tech

CORNELL TECH
HOME OF THE JACOBS INSTITUTE

OUR STUDENTS

GROWTH

- 7 in 2013
- 500 in 2021
- 1000 in 2030

PROGRAMS

Engineering
- PhD, M.Eng. in CS
- PhD, M.Eng. in ORIE
- PhD, M.Eng. in ECE
- PhD in Information Science
- PhD in Applied Math

Professional
- Johnson-Cornell Tech MBA
- Tech Law LLM

Jacobs Technion Dual Degrees
- M.S. in Connective Media
- M.S. in Health Tech
- M.S. in Urban Tech

OUR FACULTY

GROWTH

- 5 in 2013
- 38 in 2021
- 80 in 2030

STRENGTHS

- AI: ML, Vision, NLP, Robotics
- Security & Privacy, DeFi
- Mixed & Augmented Reality
- Health Tech
- Human-Centered Computing
- Tech Law, Policy, and Ethics

- Faculty joint appointments include Google, Samsung, UnitedHealth Group, and Weill Cornell Medicine

CETA CLINIC TO END TECH ABUSE
arXiv
DLI DIGITAL LIFE INITIATIVE
The Initiative for CryptoCurrencies and Contracts

ENTREPRENEURSHIP & INDUSTRY

STUDIO

- 370 companies engaged since 2014
- 35 industry practitioners have taught on campus

SPINOUTS

- > 80 startups created,
- 95% in NYC
- $157M raised (including Cornell Tech investment)
- $500M enterprise valuation

nanit
OSO
Biotia
Concertio
GITLINKS
ONETHREE BIOTECH

BROADENING PARTICIPATION

K-12

Making computer science teachable in NYC public schools:

- 5000+ students, 250+ teachers engaged
- Creating, researching and disseminating tools for teaching and learning
- NYC and national partnerships to expand scale

BREAK THROUGH TECH

Accelerating gender equality in tech:

- 3000+ students engaged in AI and Computing programs
- 100+ industry partners contributing projects, mentors, and internships

BREAK THROUGH TECH

All too familiar headlines…

Many Things Need Attention

- User interfaces (and users)
- Underlying Architecture
- Mismatch of Abstractions
- Configuration & Operation

But one issue dominates:

The code upon which we depend is full of bugs.

What’s Going Wrong?

Development processes are ineffective.

- Human code review doesn’t work.
- Analysis tools have too many false positives.

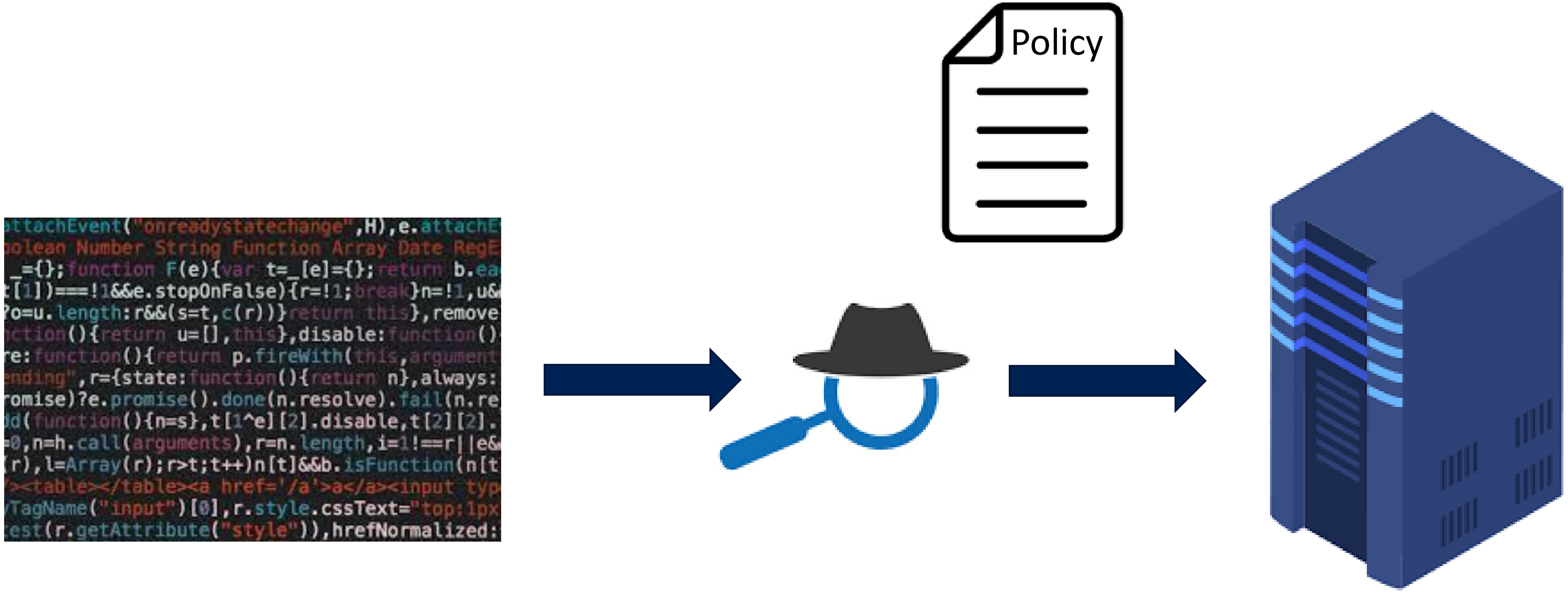
Certification processes are ineffective.

- Based on who authored, not the code itself.
- A serious threat to the viability of open source.

Automated defenses are worse than ineffective.

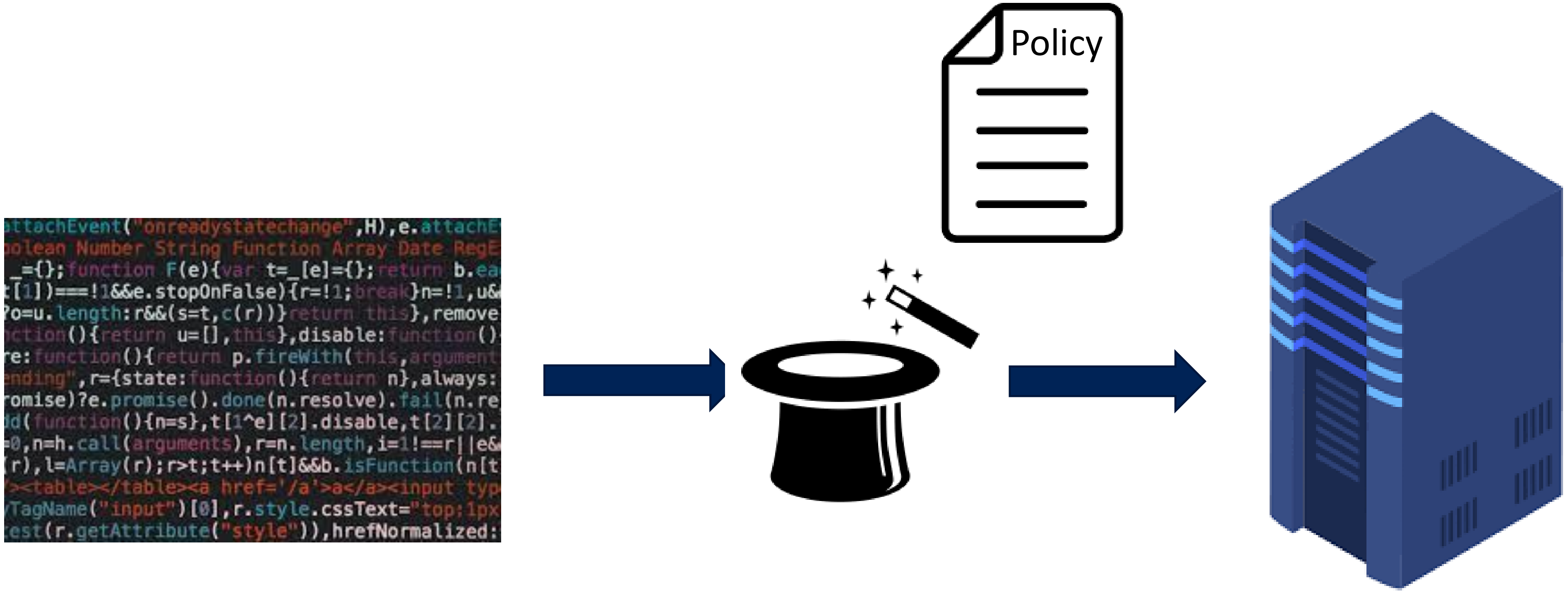
- Based on *syntax* or *provenance,* not semantics.

Ideal Architecture



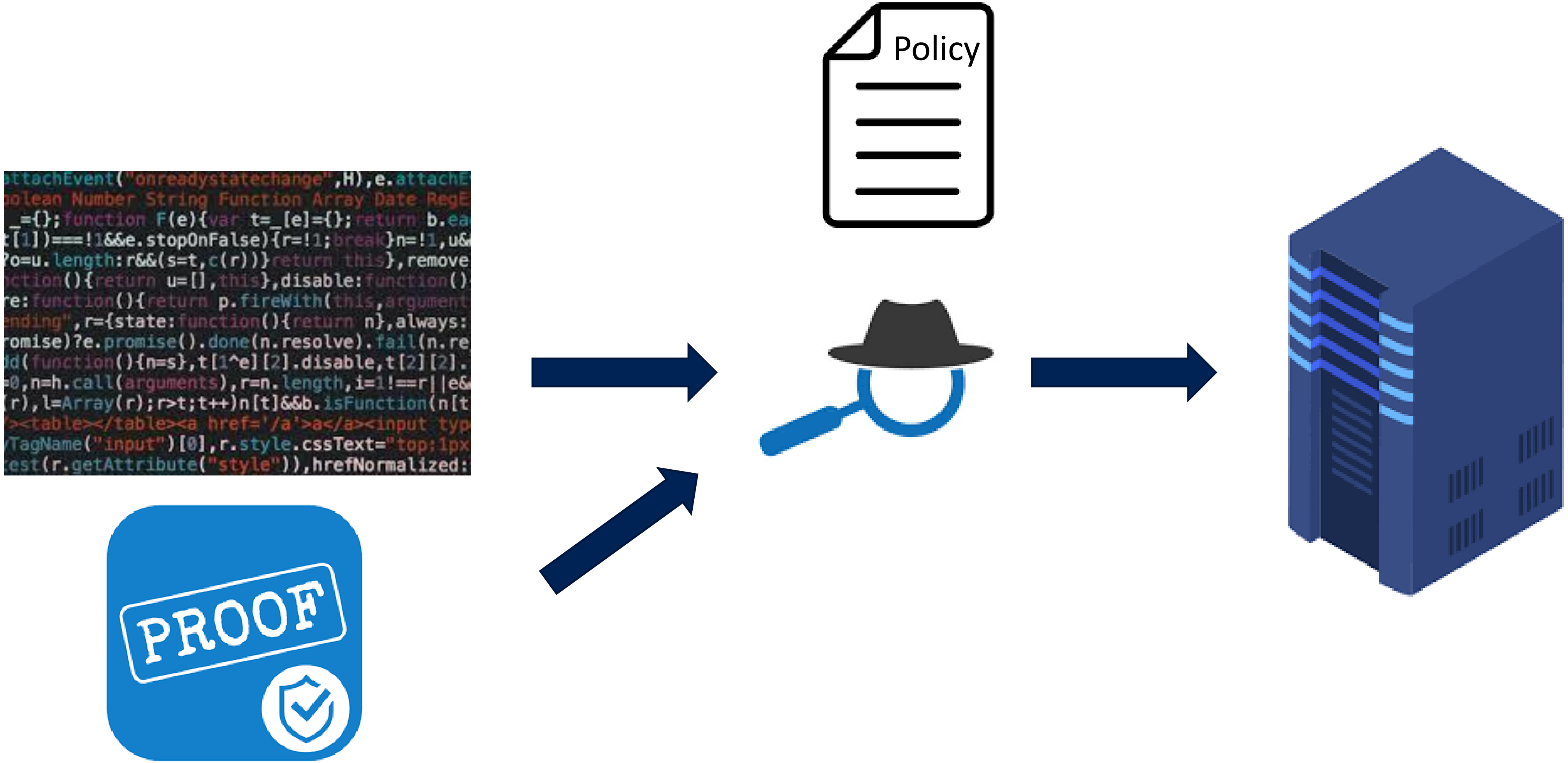
- Policies capture intended *behavior.*
- Inspector rules out any code that will violate the policy.
- The inspector is simple, trustworthy, and automatic.

Unfortunately



- It’s hard to formally capture all security requirements.
- Almost all interesting policies are (wildly) undecidable.

Shift the Burden



Formal Methods

Machine-checkable proofs enable a lot:

- Can cover all execution paths in the code.
- Can make it easier to modify code with confidence.
- Don't need to care *who* produced the code/proof.

But this is an old idea. What's changed?

Formal Methods Today

- Languages, frameworks, & logics for reasoning about code.
  - Coq, Agda, Isabelle, F*, etc.
  - Concurrent separation logic
- Formal models of real systems
  - Machines: x86, Arm, etc.
  - Languages: C, Javascript, etc.
- Proof automation: SAT & SMT solvers
  - Multiple orders of magnitude improvement over 20 years

Some Examples

- Compilers: Inria’s CompCert
- Operating Systems: NICTA’s seL4
- Crypto: Microsoft’s Everest
- Networking: Amazon’s Access Analyzer
- Hardware: MIT’s Kami

An Example Success Story

High Assurance Cyber-Military Systems (HACMS)

HACMS 18-month Program

- Clean slate software stack
  - Stability control, altitude hold, direction hold, DOS detection & response
  - GPS waypoint navigation (80%)
- Proved system-wide properties
  - System is memory safe
  - System ignores mal-formed messages
  - System ignores non-authenticated messages
  - All “good” messages will reach the controller

HACMS Evaluation

A red team was given full access to the source code for six weeks and told to break it.

They weren't able to.

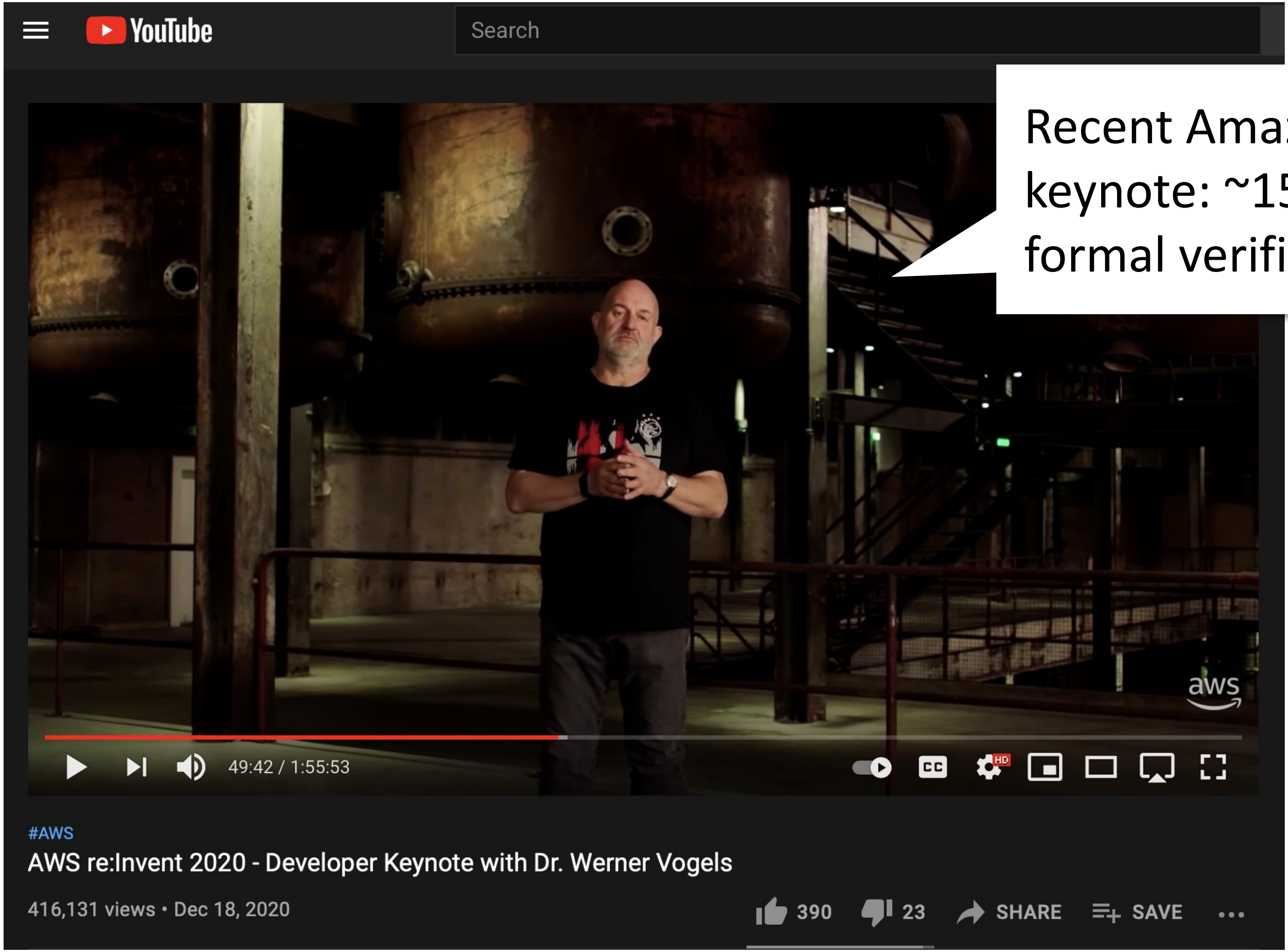
DARPA Drone Cybersecurity Software Foils Hackers in Demo

SHARE ARTICLE

Aug. 13, 2021 | By Shaun Waterman

The Defense Advanced Research Projects Agency is so confident in the hack-proof software it developed for a remote-controlled quadcopter that it invited hackers at the recent DEF CON cybersecurity convention to try to break in and take it over.

None succeeded, according to Ray Richards, program manager of DARPA's Information Innovation Office. Work on DARPA's High-Assurance Cyber Military Systems, or HACMS,



Recent Amazon CTO keynote: ~15mins on formal verification

Many Challenges

- Formalizing security policies
- Constructing & validating environment models
- Deeper automation in theorem proving
- Architectures for reducing proof burden
- Training

Many Challenges

- **Formalizing security policies**
  - Crypto world is surprisingly informal, mix assumptions
  - What’s the ”correctness” requirement for a browser?
  - What’s the policy for an ML-based system?
- Constructing & validating environment models
- Deeper automation in theorem proving
- Architectures for reducing proof burden
- Training

Many Challenges

- Formalizing security policies
- Constructing & validating environment models
  - For example, a CPU or network card or firewall?
  - How do we test these models?
  - What level of abstraction (e.g., Spectre/Rowhammer?)
- Deeper automation in theorem proving
- Architectures for reducing proof burden
- Training

Many Challenges

- Formalizing security policies
- Constructing & validating environment models
- **Deeper automation in theorem proving**
  - seL4 took 20 person-years to prove secure
  - SMT only handles quantifier-free fragment
  - Not yet taking full advantage of cluster-scale compute
  - Can ML be applied to synthesizing proofs?
- Architectures for reducing proof burden
- Training

Many Challenges

- Formalizing security policies
- Constructing & validating environment models
- Deeper automation in theorem proving
- **Architectures for reducing proof burden**
  - Encapsulate and test (e.g., register allocator)
  - Compositional abstractions (e.g., CerticOS)
  - Reusable libraries/models
- Training

Many Challenges

- Formalizing security policies
- Constructing & validating environment models
- Deeper automation in theorem proving
- Architectures for reducing proof burden
- Training: need proof engineering
  - Constructing & maintaining proofs is still hard work
  - Few universities teach this well
  - We need a new field of proof engineering!

Only the Beginning

Formal methods can help with the code bugs, but fundamentally leaves these hard problems:

- User interfaces (and users)
- Underlying Architecture
- Mismatch of Abstractions
- Configuration & Operation