Carnegie Mellon University
Software Engineering Institute

Using AI to Build More Secure Software

Abstractions II Conference
Aug 23, 2019

Dr. Mark Sherman
Technical Director, Cyber Security Foundations

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

DM19-0825

The SEI is a DoD R&D Federally Funded Research and Development Center

Established in 1984 at Carnegie Mellon University

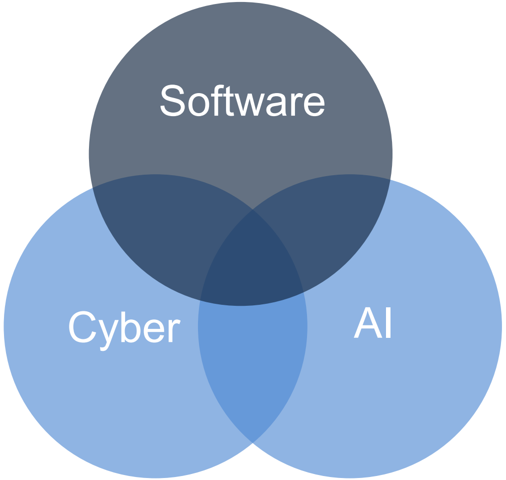
~650 employees (ft + pt), of whom about 70% are engaged in technical work

Initiated CERT cybersecurity program in 1988: the birthplace of cybersecurity

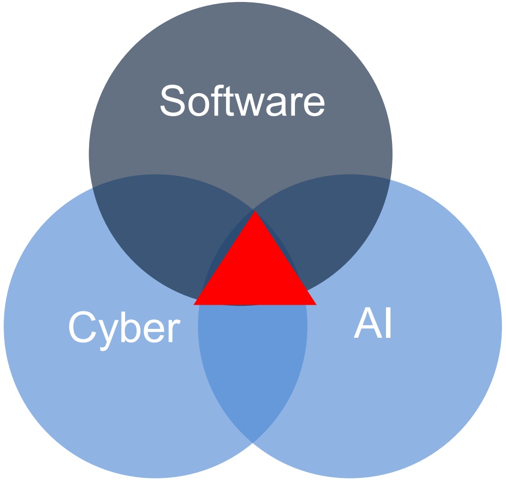
Offices in Pittsburgh and DC, with several locations near customer facilities

~$140M in annual funding

SEI Strategic Framework



SEI Strategic Framework – Today's Focus



Software Cost and Vulnerability Threaten Military Capability

Finding and fixing software problems late in the acquisition lifecycle drives up cost and delays delivery

Latent cyber vulnerabilities and those exposed during operations or due to underlying dependencies put missions at risk

Statistically, a 10M LOC Weapons Platform written in C will be delivered with 280 – 1,400 exploitable vulnerabilities

F-15 Fighter Jet Hacked

Image: A U.S. Air Force F-15E Strike Eagle. (U.S. Air Force photo by Senior Airman Erin Trowe)

THE CYBERSECURITY 202: HACKERS JUST FOUND SERIOUS VULNERABILITIES IN A U.S. MILITARY FIGHTER JET

LAS VEGAS — In a Cosmopolitan hotel suite 16 stories above the Def Con cybersecurity conference this weekend, a team of highly vetted hackers tried to sabotage a vital flight system for a U.S. military fighter jet. And they succeeded.

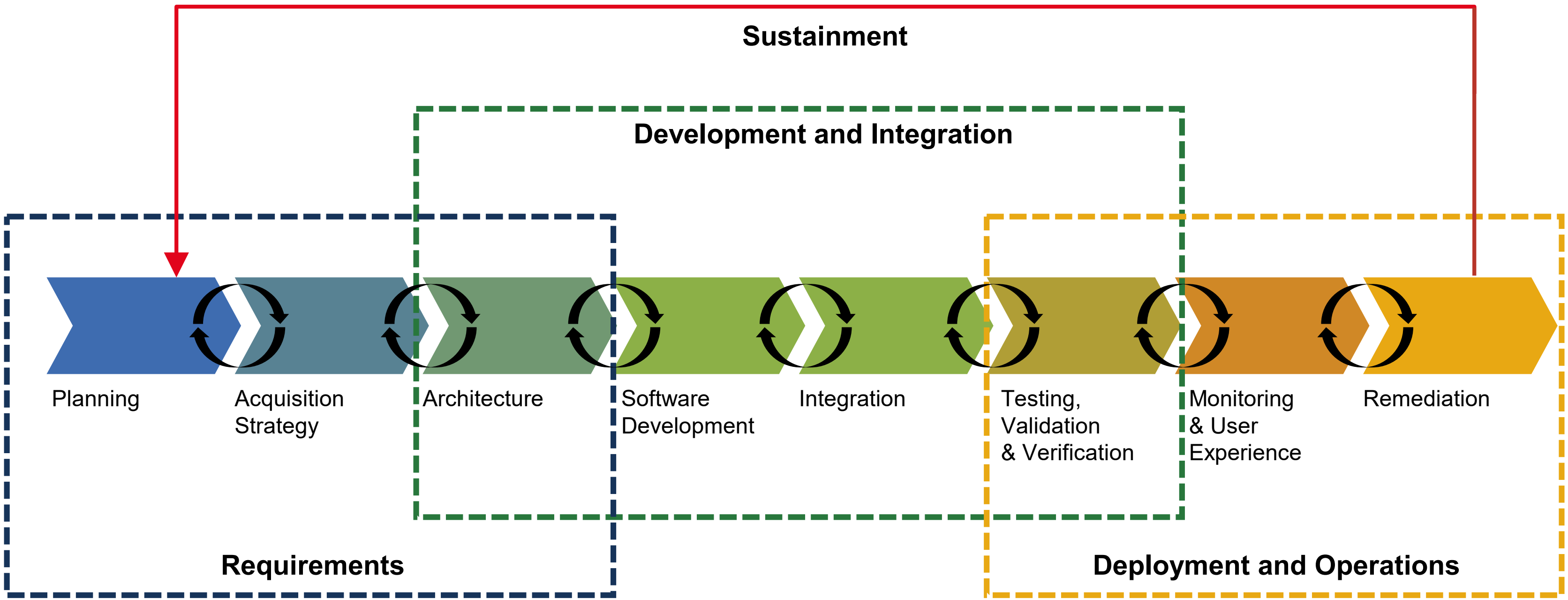
It was the first time outside researchers were allowed physical access to the critical F-15 system to search for weaknesses. And after two long days, the seven hackers found a mother lode of vulnerabilities that — if exploited in real life — could have completely shut down the Trusted Aircraft Information Download Station, which collects reams of data from video cameras and sensors while the jet is in flight.

Will Roper, a top U.S. Air Force acquisitions executive, told the Washington Post: "there are millions of lines of code that are in all of our aircraft and if there's one of them that's flawed, then a country that can't build a fighter to shoot down that aircraft might take it out with just a few keystrokes.“

Joseph Marks, Aug 14, 2019

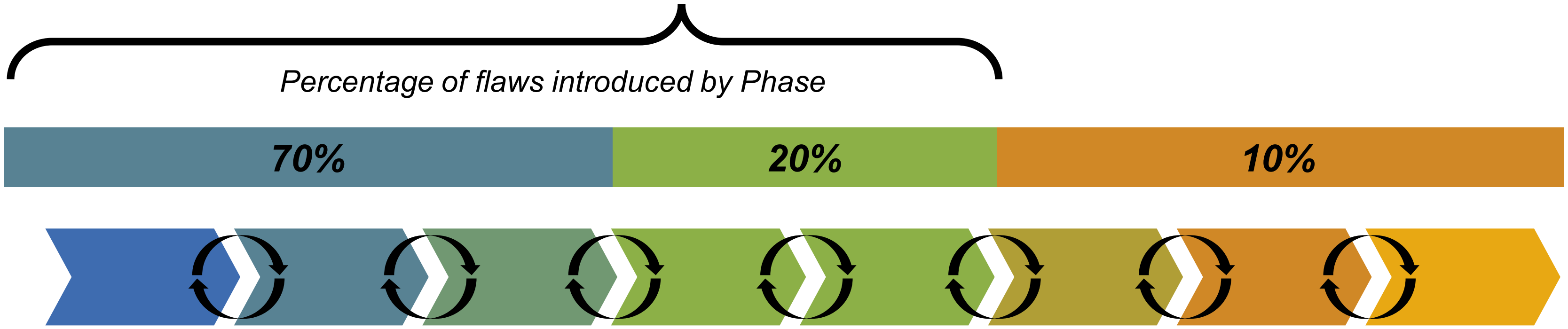
Source: https://https://www.washingtonpost.com/news/powerpost/paloma/the-cybersecurity-202/2019/08/14/the-cybersecurity-202-hackers-just-found-serious-vulnerabilities-in-a-u-s-military-fighter-jet/5d53111988e0fa79e5481f68/

Fixing Problems Late Drives Costs, Delays Deployment


Sustainment
Development and Integration
Planning
Acquisition Strategy
Architecture
Software Development
Integration
Testing, Validation & Verification
Monitoring & User Experience
Remediation
Requirements
Deployment and Operations

Fixing Problems Late Drives Costs, Delays Deployment

Software problems that drive costs
are introduced early in the lifecycle . . .



Percentage of flaws introduced by Phase

70% ***20%*** ***10%***

Planning

Acquisition Strategy

Architecture

Software Development

Integration

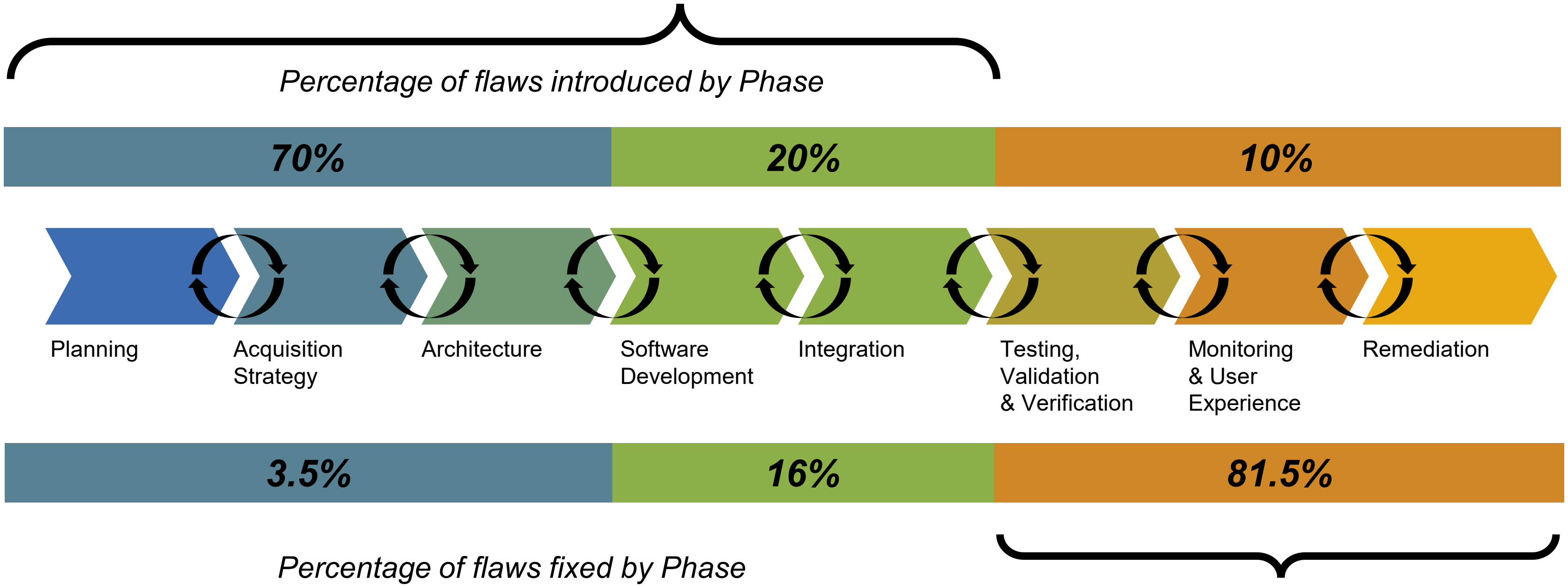
Testing, Validation & Verification

Monitoring & User Experience

Remediation

Fixing Problems Late Drives Costs, Delays Deployment

Software problems that drive costs
are introduced early in the lifecycle . . .



Percentage of flaws introduced by Phase

70% ***20%*** ***10%***

Planning

Acquisition Strategy

Architecture

Software Development

Integration

Testing, Validation & Verification

Monitoring & User Experience

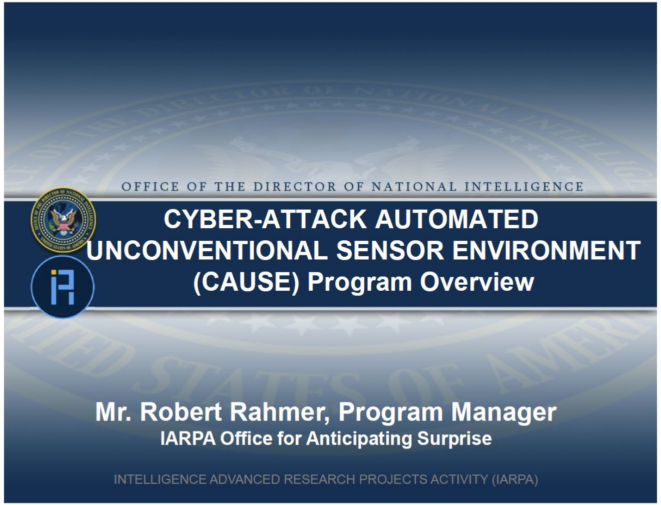
Remediation

3.5% ***16%*** ***81.5%***

Percentage of flaws fixed by Phase

. . . But discovered late, increasing cost,
vulnerability, and schedule impact

Predicting Threats – IARPA CAUSE



OFFICE OF THE DIRECTOR OF NATIONAL INTELLIGENCE

CYBER-ATTACK AUTOMATED UNCONVENTIONAL SENSOR ENVIRONMENT (CAUSE) Program Overview

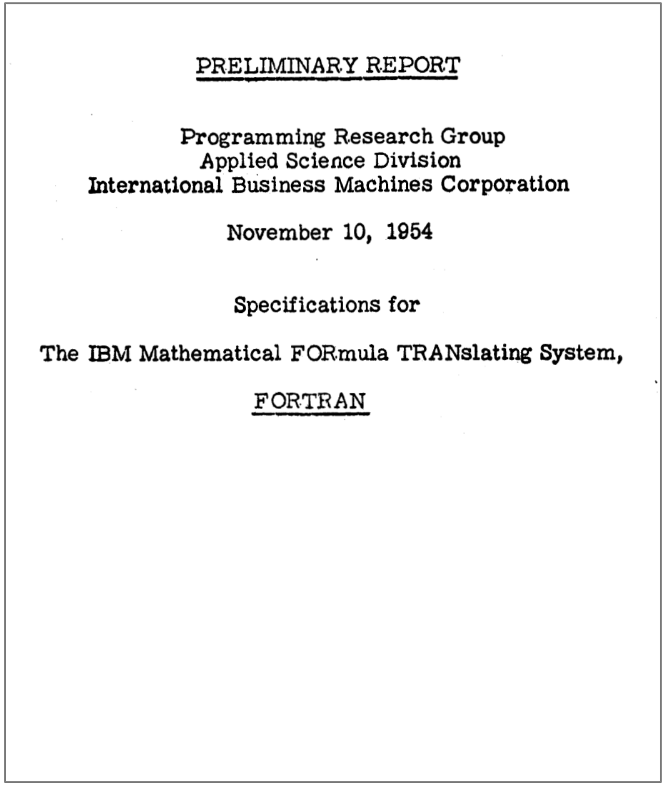
Mr. Robert Rahmer, Program Manager
IARPA Office for Anticipating Surprise

INTELLIGENCE ADVANCED RESEARCH PROJECTS ACTIVITY (IARPA)

- Identify and evaluate unconventional and technical indicators in the earlier phases of cyber attacks that are leading indicators of later stages of the attack.
- Create highly efficient algorithms that will process massive data streams from diverse data sets to extract signals from noisy data.
- Create techniques to fuse traditional technical indicator sensor data and alternate unconventional indicator data sources to develop automated probabilistic warnings.
- Identify and evaluate techniques that enable sharing of disparate threat contextual information and indicators among multiple organizations and security professionals to forecast an attack.

Sources: IARPA, Cyber-attack Automated Unconventional Sensor Environment (CAUSE), https://www.iarpa.gov/index.php/research-programs/cause https://www.iarpa.gov/images/files/programs/cause/CAUSE_Proposers_Day_Briefing.pdf

AI in Automatic Programming – The Beginning



PRELIMINARY REPORT

Programming Research Group
Applied Science Division
International Business Machines Corporation

November 10, 1954

Specifications for

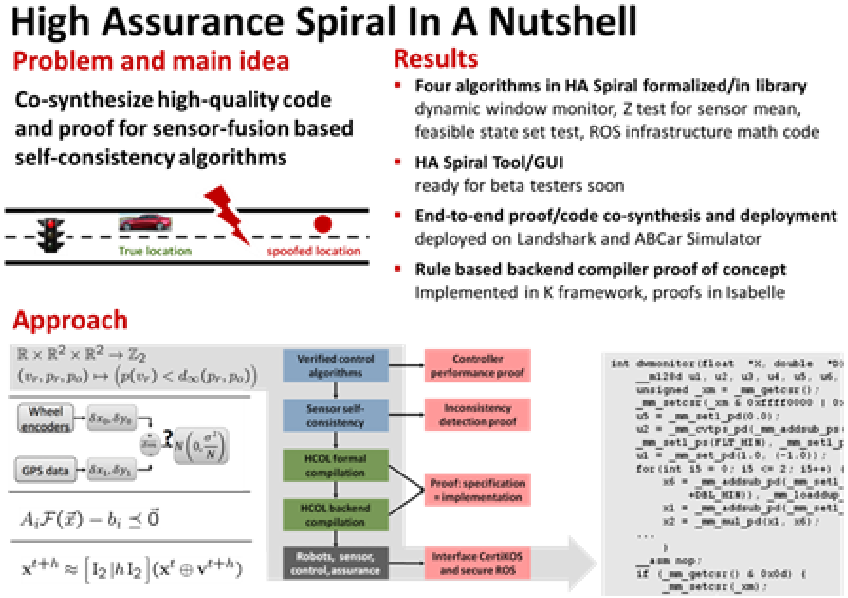
The IBM Mathematical FORmula TRANslating System,

FORTRAN

“The IBM Mathematical Formula Translating System or briefly, FORTRAN, will comprise a large set of programs to enable the IBM 704 to accept a concise formulation of a problem in terms of a mathematical notation and to produce automatically a high speed 704 program for the solution of the problem.”

Source: J.W. Backus, H. Herrick and I. Ziller,
https://archive.computerhistory.org/resources/text/Fortran/102679231.05.01.acc.pdf

AI in Automatic Programming: Generating Coded thru Search – High Assurance SPIRAL



High Assurance Spiral In A Nutshell

Problem and main idea

Co-synthesize high-quality code and proof for sensor-fusion based self-consistency algorithms

True location
spoofed location

Results

- Four algorithms in HA Spiral formalized/in library
  dynamic window monitor, Z test for sensor mean, feasible state set test, ROS infrastructure math code
- HA Spiral Tool/GUI
  ready for beta testers soon
- End-to-end proof/code co-synthesis and deployment
  deployed on Landshark and ABCar Simulator
- Rule based backend compiler proof of concept
  Implemented in K framework, proofs in Isabelle

Approach

$\mathbb{R} \times \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{Z}_2$
$(v_r, p_r, p_o) \mapsto \left(p(v_r) < d_\infty(p_r, p_o)\right)$

$A_i \mathcal{F}(\vec{x}) - b_i \preceq \vec{0}$

$\mathbf{x}^{t+h} \approx \left[\mathrm{I}_2 \,|\, h\, \mathrm{I}_2 \right] (\mathbf{x}^t \oplus \mathbf{v}^{t+h})$

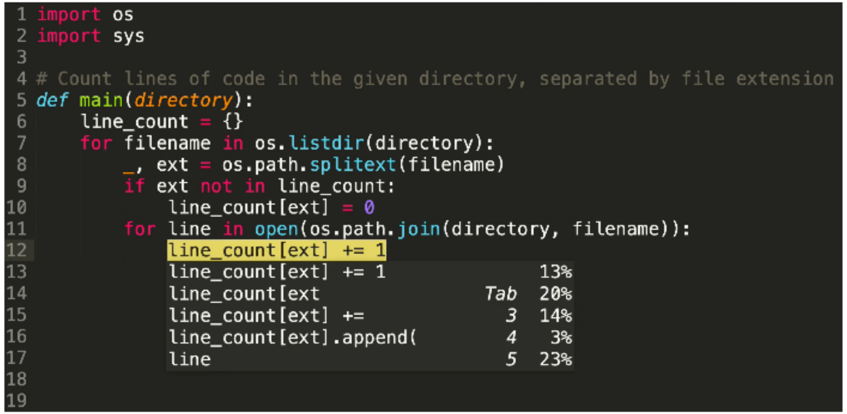
Verified control algorithms
Sensor self-consistency
HCOL formal compilation
HCOL backend compilation
Robots, sensor, control, assurance
Controller performance proof
Inconsistency detection proof
Proof: specification = implementation
Interface CertiKOS and secure ROS

```
int dwmonitor(float  *X, double  *D
    __m128d u1, u2, u3, u4, u5, u6,
    unsigned _xm = _mm_getcsr();
    _mm_setcsr(_xm & 0xffff0000 | 0x
    u5 = _mm_set1_pd(0.0);
    u2 = _mm_cvtps_pd(_mm_addsub_ps(
    _mm_set1_ps(FLT_MIN), _mm_set1_p
    u1 = _mm_set_pd(1.0, (-1.0));
    for(int i5 = 0; i5 <= 2; i5++) {
        x6 = _mm_addsub_pd(_mm_set1_
            +DBL_MIN)), _mm_loaddup_
        x1 = _mm_addsub_pd(_mm_set1_
        x2 = _mm_mul_pd(x1, x6);
    ...
        }
    __asm nop;
    if (_mm_getcsr() & 0x0d) {
        _mm_setcsr(_xm);
```

"*High Assurance SPIRAL* aims to solve the last mile problem for the synthesis of high assurance implementations of controllers for vehicular systems that are executed in todays and future embedded and high performance embedded system processors."

Sources: Franz Franchetti, José M. F. Moura, Manuela Veloso, Andre Platzer, Soummya Kar, David Padua, Jeremy Johnson, Mike Franusich, High Assurance Spiral: Scalable and Performance Portable Domain-Specific Control System Synthesis, https://users.ece.cmu.edu/~franzf/hacms.htm; http://www.spiral.net/

Using AI For Autocompletion



```
import os
import sys

# Count lines of code in the given directory, separated by file extension
def main(directory):
    line_count = {}
    for filename in os.listdir(directory):
        _, ext = os.path.splitext(filename)
        if ext not in line_count:
            line_count[ext] = 0
        for line in open(os.path.join(directory, filename)):
            line_count[ext] += 1
            line_count[ext] += 1          13%
            line_count[ext           Tab  20%
            line_count[ext] +=         3  14%
            line_count[ext].append(    4   3%
            line                       5  23%
```

Safe, correct code could be written incrementally

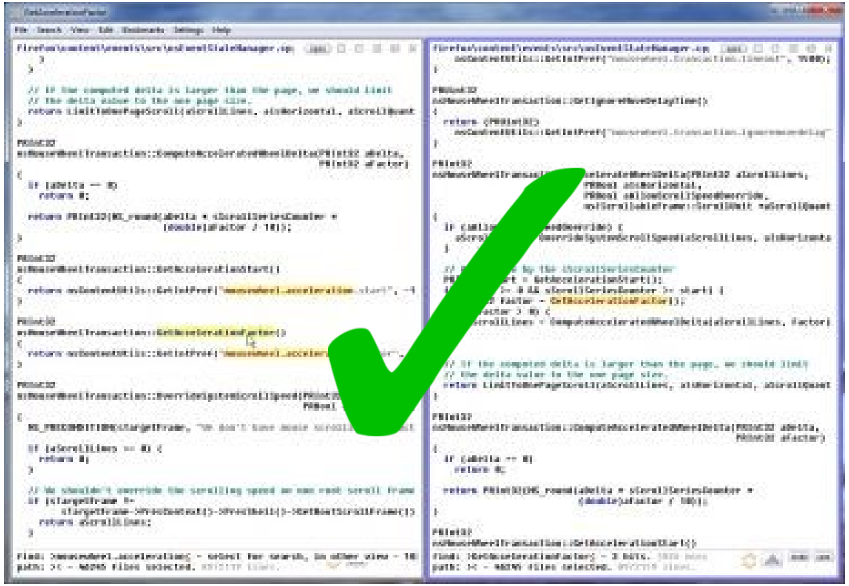
- Using n-grams
- Using deep learning (Generative Pretrained Transformer 2)

Sources:
E. Schutte, Autocomplete from StackOverflow, 2016, https://emilschutte.com/stackoverflow-autocomplete/

(Jacob Jackson) TabNine, “Autocompletion with deep learning,” July 18, 2019, https://tabnine.com/blog/deep

L. Tung, “New tool promises to turbo-charge coding in major programming languages,” July 25, 2019, https://www.zdnet.com/article/new-tool-promises-to-turbo-charge-coding-in-major-programming-languages/

Finding Programming Vulnerabilities – Source Code as Natural Language



Analyze Source Code for Insecure Coding

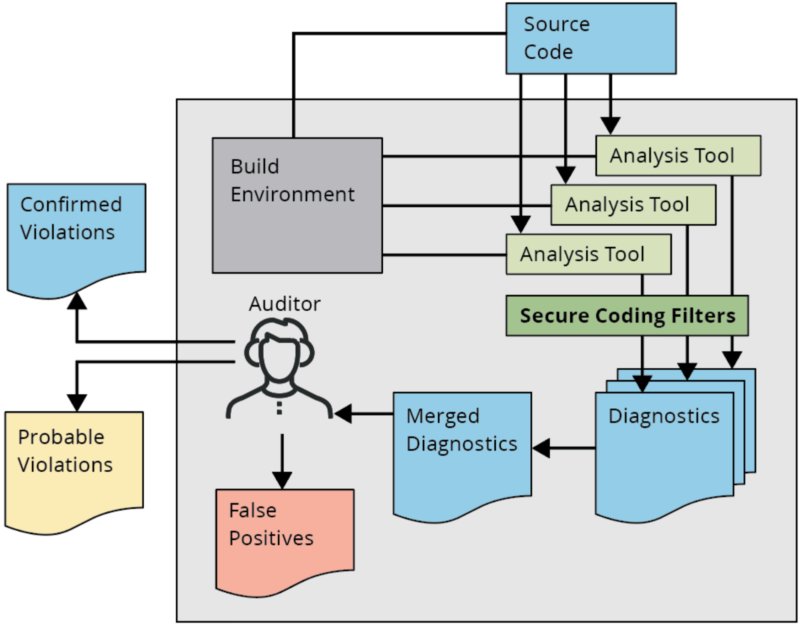
- Supplements Compiler-style Checking
- Treats Programs Like Natural Language

Sources: Carson D. Sestili, William S. Snavely, Nathan M. VanHoudnos, Towards security defect prediction with AI, Sep 12, 2018, https://arxiv.org/abs/1808.09897

Song Wang, Taiyue Liu, and Lin Tan. 2016. Automatically learning semantic features for defect prediction. In *Proceedings of the 38th International Conference on Software Engineering* (ICSE '16). ACM, New York, NY, USA, 297-308. DOI: https://doi.org/10.1145/2884781.2884804

Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. 2019. code2vec: learning distributed representations of code. Proc. ACM Program. Lang. 3, POPL, Article 40 (January 2019), 29 pages. DOI: https://doi.org/10.1145/3290353

Combining Multiple Tools With AI To Find Source Code Flaws – SCALe



Using AI and Machine Learning to Combine Tool and Environmental Data

- Multiple static code analyzers
- Multiple environmental features
- Multiple classification techniques

Source: Lori Flynn, SCALe: A Tool for Managing Output from Static Analysis Tools, Sept 24, 2018, https://insights.sei.cmu.edu/sei_blog/2018/09/scale-a-tool-for-managing-output-from-static-code-analyzers.html ; Lori Flynn, Automating Static Analysis Alert Handling with Machine Learning, MIT Lincoln Labs Cyber Security, Exploitation and Operations Workshop, June 19, 2018;

Using AI to Drive Test Inputs – Fuzzing

“Fuzzing:” Generating and Testing Random Inputs

Original: Random or Deterministic

Now: Use AI to Guide Generation of Sample Inputs

Sources: A. Householder, Announcing CERT Basic Fuzzing Framework Version 2.8, Oct. 5, 2016, https://insights.sei.cmu.edu/cert/2016/10/announcing-cert-basic-fuzzing-framework-bff-28.html

G. Yan, J. Lu, Z. Shu ; Y. Kucuk, “ExploitMeter: Combining Fuzzing with Machine Learning for Automated Evaluation of Software Exploitability,” 2017 IEEE Symposium on Privacy-Aware Computing (PAC), 1-4 Aug. 2017, https://doi.org/10.1109/PAC.2017.10

D. She, K. Pei, D. Epstein, J. Yang, B. Ray, S. Jana, “NEUZZ: Efficient Fuzzing with Neural Program Smoothing,” 40th IEEE Symposium on Security and Privacy, May 20--22, 2019, San Francisco, CA, USA, https://arxiv.org/pdf/1807.05620.pdf

Using AI to Improve Penetration Testing

Variety and combination of manual techniques can be executed by an AI system

- AI planning using an attack graph against attack surfaces
- Markov Decision Process (or Partially Observable Markvo Decision Process) over application state
- Reinforcement learning

Sources:

K. Durkota and V. Lisy, “Computing Optimal Policies for Attack Graphs with Action Failures and Costs,” Conference: Proceedings of the 7th Starting AI Researchers' Symposium (STAIRS), December 2013, https://www.researchgate.net/profile/Karel_Durkota/publication/273640839_Computing_Optimal_Policies_for_Attack_Graphs_with_Action_Failures_and_Costs

C. Sarraute, O. Buffet, and J. Hoffmann, “POMDPs Make Better Hackers: Accounting for Uncertainty in Penetration Testing,” AAAI, 2012, https://arxiv.org/pdf/1307.8182

J. Schwartz, “Autonomous Penetration testing using Reinforcement Learning, Nov 16, 2018, https://arxiv.org/ftp/arxiv/papers/1905/1905.05965.pdf

Using AI for Automated Cyber Red Teaming

Red teaming simulates an attack across a system to evaluate the cybersecurity of a mission

Typically depends on having

- Cyber ontology of mission and system elements supporting mission
- Attack trees
- Mission priorities
- Planners: state-space planners, planning graph planners and hierarchical task network-based planners

Sources:

S. Randhawa, et al, “Mission-Centric Automated Cyber Red Teaming,” ARES 2018, Proceedings of the 13th International Conference on Availability, Reliability and Security, August 27–30, 2018, Hamburg, Germany, https://www.researchgate.net/publication/327005899_Mission-Centric_Automated_Cyber_Red_Teaming

S. Upton, et al, "Breaking blue: Automated red teaming using evolvable simulations," Proceedings of Genetic and Evolutionary Computation Conference 2004, 2004, http://gpbib.cs.ucl.ac.uk/gecco2004/WMSA015.pdf

J. Yuen, "Automated Cyber Red Teaming," DTIC Document2015, https://apps.dtic.mil/docs/citations/ADA618584

Automated Program Repair – DARPA Cyber Grand Challenge

"Mayhem" demonstrated automated cyber defense

- Detect attack on program
- Analyze changes to program
- Deploy updated software

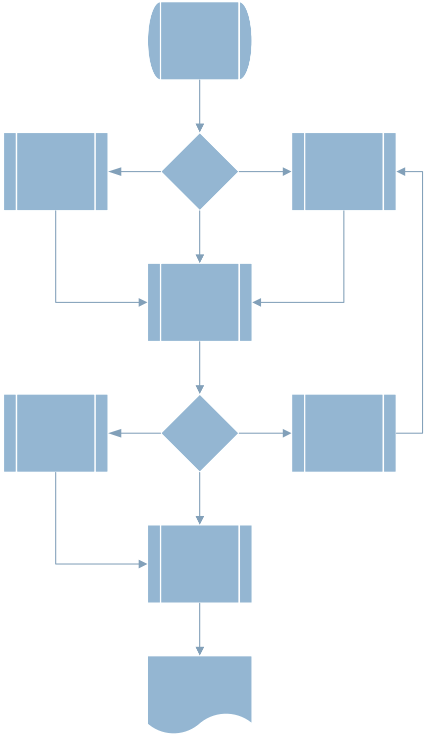
Source: DARPA, "Mayhem" Declared Preliminary Winner of Historic Cyber Grand Challenge, Aug 4, 2016, https://www.darpa.mil/news-events/2016-08-04

AI Supporting Judgement – IBM Watson to Improve Assurance

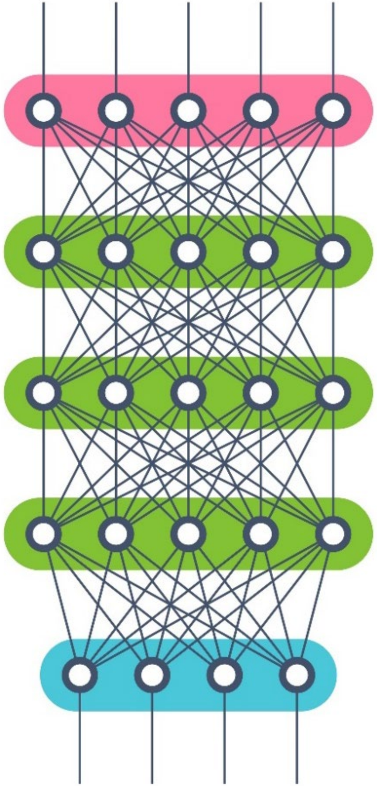
- Acquisition programs generate voluminous documentation
- Assurance is based on assembling and reviewing relevant evidence from documents
- Finding appropriate evidence or explanations can be challenging
- SEI Proof of Concept

Source: Mark, Sherman, Verifying Software Assurance with IBM's Watson, https://www.youtube.com/watch?v=aW3497xhypY, Sep 11, 2017

Machine Learning is a Different Style of Programming

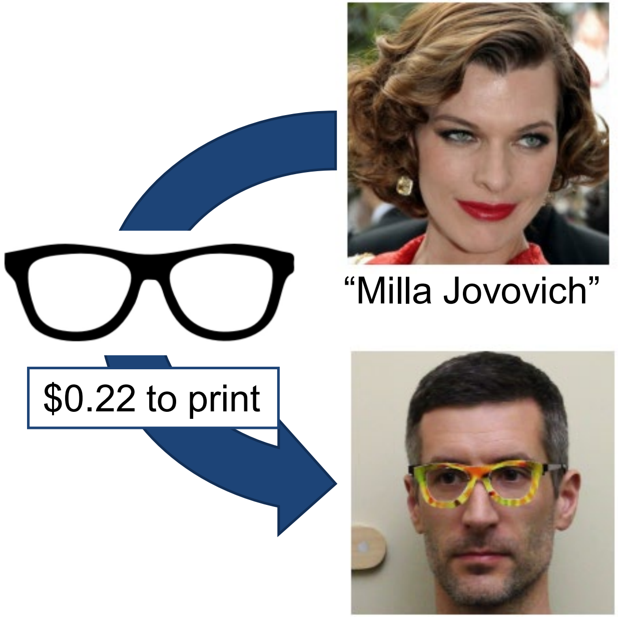


VS



AI Attacks Are Different

Pixel Manipulation

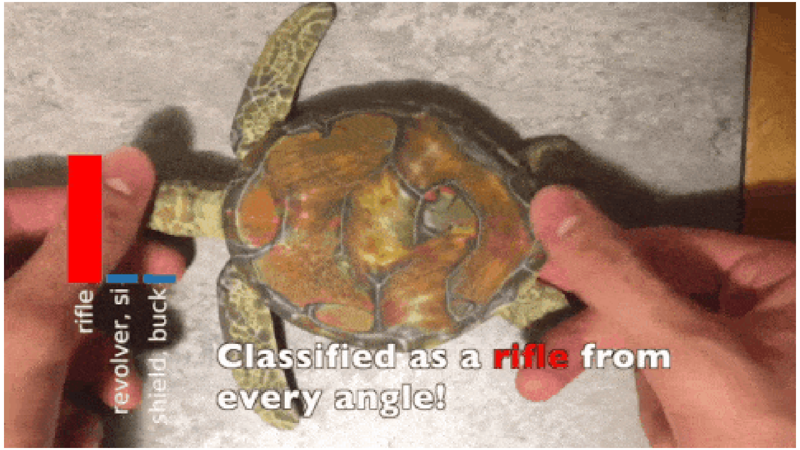


$0.22 to print

“Milla Jovovich”

“Milla Jovovich”

Feature Differentiation



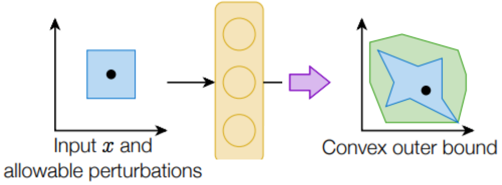
Source: Athalye, A., Engstrom, L., Ilyas, A., & Kwok, K. (2017, July 24). *Synthesizing Robust Adversarial Examples*. *arXiv [cs.CV]*. Retrieved from http://arxiv.org/abs/1707.07397

Source: Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. 2016. Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (CCS '16). ACM, New York, NY, USA, 1528-1540. DOI: https://doi.org/10.1145/2976749.2978392

Some Technical Approaches for Defending AI Systems

Training Defenses

Wong & Kolter (2017) output bound



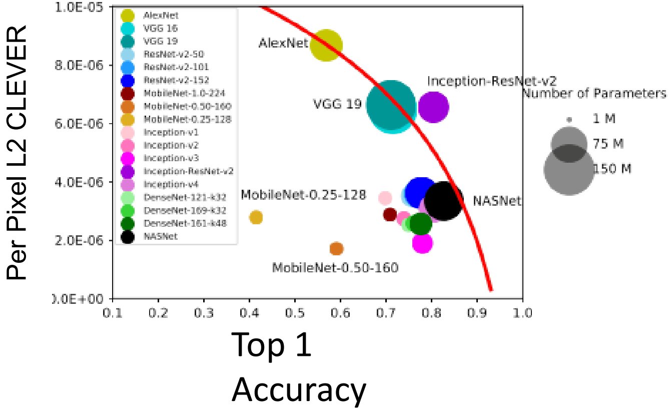
Causal Defenses

Tsipras et al. (2018) adversarial data augmentation

Turtle → Bird

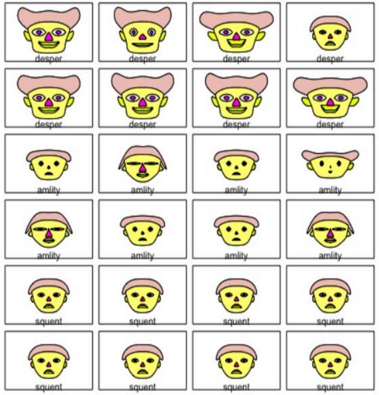
Engineering Defenses

Su et al. (2018) empirically demonstrates robustness/accuracy trade off in ImageNet models

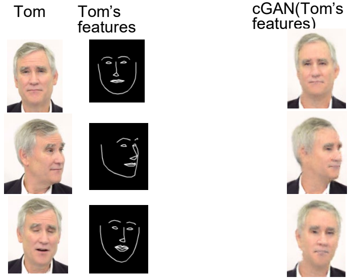


Source: Wong, E., & Kolter, J. Z. (2017). Provable defenses against adversarial examples via the convex outer adversarial polytope. ArXiv:1711.00851 [Cs, Math]. Retrieved from http://arxiv.org/abs/1711.00851; Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., & Madry, A. (2018). Robustness May Be at Odds with Accuracy. ArXiv:1805.12152 [Cs, Stat]. Retrieved from http://arxiv.org/abs/1805.12152; Su, D., Zhang, H., Chen, H., Yi, J., Chen, P.-Y., & Gao, Y. (2018). Is Robustness the Cost of Accuracy? – A Comprehensive Study on the Robustness of 18 Deep Image Classification Models. ArXiv:1808.01688 [Cs]. Retrieved from http://arxiv.org/abs/1808.01688; Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I.P. Rubinstein, and J. D. Tygar. 2011. Adversarial machine learning. In Proceedings of the 4th ACM workshop on Security and artificial intelligence (AISec '11). ACM, New York, NY, USA, 43-58. DOI=http://dx.doi.org/10.1145/2046684.2046692

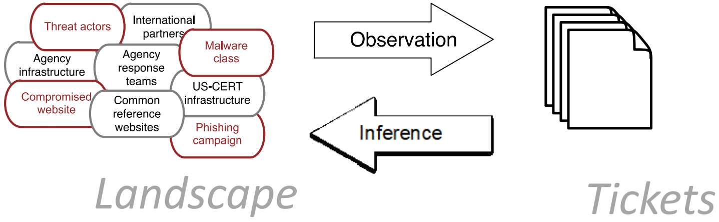
AI is Playing an Increasing Role in Cybersecurity



Classifying Malware



Spotting Deep Fakes



Detecting Campaigns

- Detecting misinformation
- Spotting command and control paths
- Cyber training

- Technical debt detection
- Satellite image recognition
- Insider threat detection

Summary: Using AI to Build More Secure Software

Problem: The Need to Build Secure Software

Threat Analysis: What To Protect Against

Code Development: Assisting Programmers to Build More Secure Software

Building AI Systems Securely: Next Generation of Software Face New Attacks

Contact Us

Carnegie Mellon University
Software Engineering Institute
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
412-268-5800
888-201-4479
info@sei.cmu.edu
www.sei.cmu.edu