Using AI to Build More Secure Software
Abstractions II Conference
Aug 23, 2019

Dr. Mark Sherman
Technical Director, Cyber Security Foundations
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
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.
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

Fixing Problems Late Drives Costs, Delays Deployment
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  Architecture  Software  Development  Integration  Testing,  Validation  &  Verification  Monitoring  &  User  Experience  Remediation

Percentage of flaws introduced by Phase

70%  20%  10%

Planning  Acquisition  Architecture  Software  Development  Integration  Testing,  Validation  &  Verification  Monitoring  &  User  Experience  Remediation

Software problems that drive costs are introduced early in the lifecycle . . .
Fixing Problems Late Drives Costs, Delays Deployment

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

**Percentage of flaws introduced by Phase**

- **Planning**: 70%
- **Acquisition Strategy**: 20%
- **Architecture**: 10%
- **Software Development**: 3.5%
- **Integration**: 16%
- **Testing, Validation & Verification**: 81.5%
- **Monitoring & User Experience**: 81.5%
- **Remediation**: 81.5%

**Percentage of flaws fixed by Phase**

- **Planning**: 3.5%
- **Acquisition Strategy**: 16%
- **Architecture**: 81.5%
- **Software Development**: 81.5%
- **Integration**: 81.5%
- **Testing, Validation & Verification**: 81.5%
- **Monitoring & User Experience**: 81.5%
- **Remediation**: 81.5%

. . . But discovered late, increasing cost, vulnerability, and schedule impact
Predicting Threats – IARPA CAUSE

- 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.

“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.”

“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 today’s and future embedded and high performance embedded system processors.”
Using AI For Autocompletion

Safe, correct code could be written incrementally

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

Sources:


© 2019 TabNine, See https://tabnine.com/eula

```python
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)[1]
        if ext not in line_count:
            line_count[ext] = 0
        for line in open(os.path.join(directory, filename)).readlines():
            if line.startswith('def '):
                line_count[ext] += 1
            else:
                if line.strip() == 'import os':
                    line_count['.py'] += 1
                else:
                    line_count[line.strip()]

if __name__ == '__main__':
    main('.')
```

© 2019 TabNine, See https://tabnine.com/eula
Finding Programming Vulnerabilities – Source Code as Natural Language

Analyze Source Code for Insecure Coding

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

Sources:
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

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:


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 Markov Decision Process) over application state
- Reinforcement learning

Sources:


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:


Automated Program Repair – DARPA Cyber Grand Challenge

“Mayhem” demonstrated automated cyber defense

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

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

Machine Learning is a Different Style of Programming

VS
AI Attacks Are Different

Pixel Manipulation

“Milla Jovovich”

$0.22 to print

Feature Differentiation

“Milla Jovovich”


Some Technical Approaches for Defending AI Systems

Training Defenses
Wong & Kolter (2017) output bound

Causal Defenses
Tsipras et al. (2018) adversarial data augmentation

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

AI is Playing an Increasing Role in Cybersecurity

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

Spotting Deep Fakes
- Technical debt detection
- Satellite image recognition
- Insider threat detection

Detecting Campaigns

Landscape
- Observation
- Inference
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