Three Software Innovations that DoD Needs Now

Jeff Boleng, Sam Procter, Nathan VanHoudnos,
Lena Pons, Robert Schiela

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Document Markings

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.

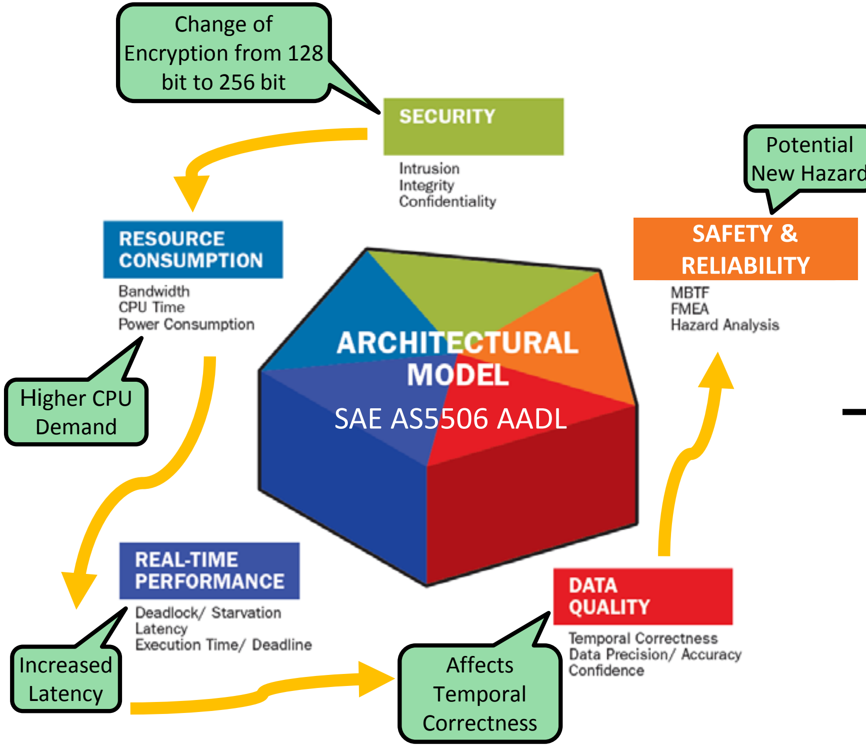
The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

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.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

DM18-0334

Virtual Integration: AADL as a “Single Source of Truth”



Model-Based Engineering for System Design

Build models of components, combine them into a unified system, and verify that everything fits before the system is built: *Integrate-then-Build*

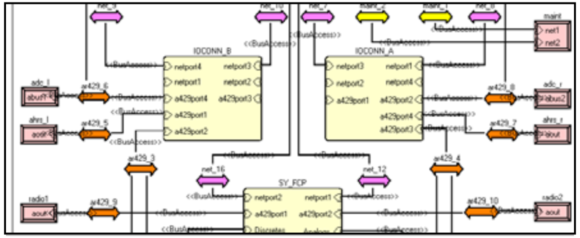
Detect Issues Early, Save Money and Effort

Studies show most system defects are created early – in the design and architecture stages – but fixed late. Defects caught later are more costly in terms of both budget and schedule overruns.

Multiple Analyses, One Model

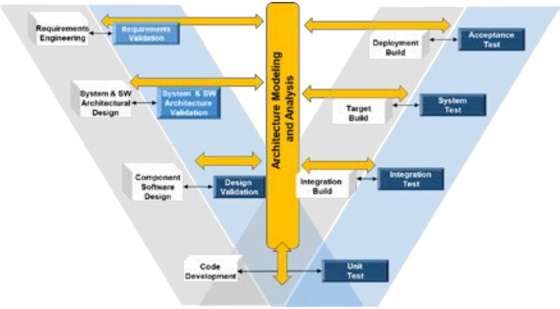
A single AADL model can support multiple analyses. Custom analyses are easy to implement, or use outputs of preferred tools to annotate AADL model

AADL Success Stories



Wheel Braking System

- Example used in SAE standardization efforts (ARP 4761 & AIR61160)
- AADL source publically available on github
- Used in ongoing safety research



System Architecture Virtual Integration

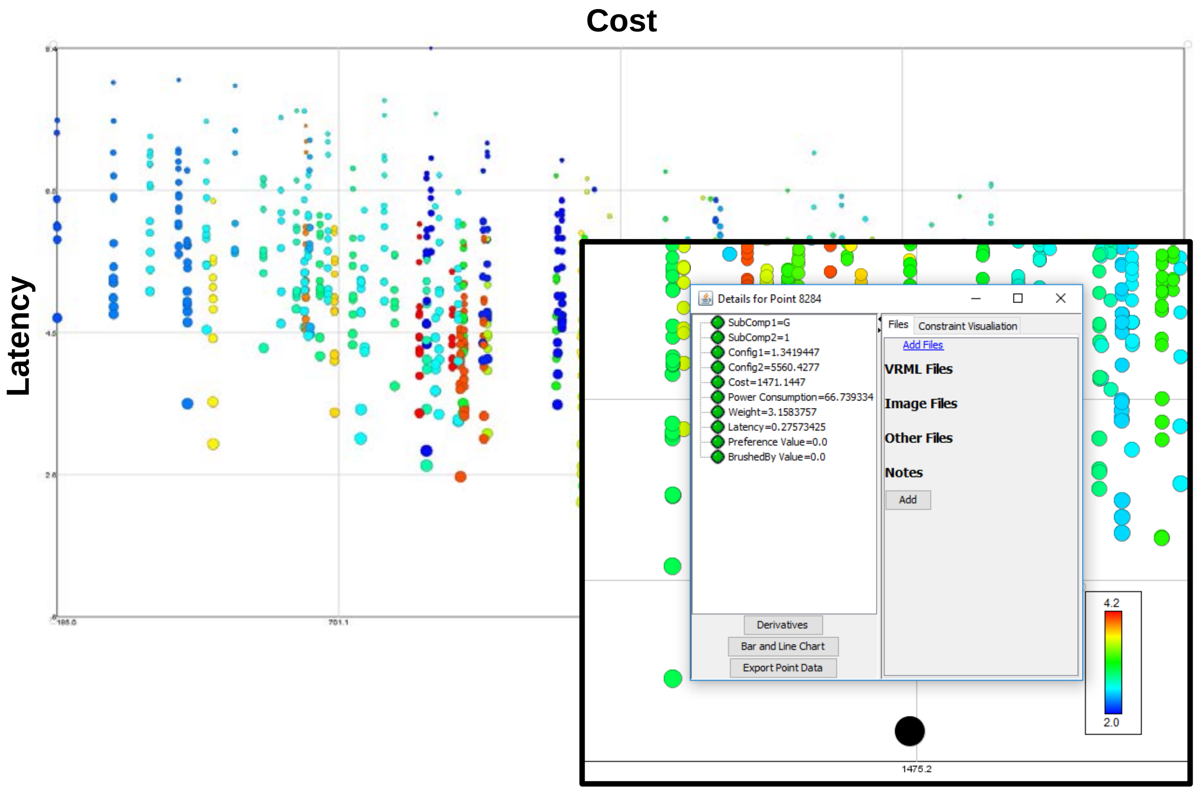
- “Incremental Validation, Continuous Integration”
- Pays for itself in commercial development

Image adapted from loonwerks.com

HACMS: Strong Security

- Secure drone and helicopter developed using AADL, seL4 & other tech
- Resistant to weeks of red-team attacks, even with source code

Guided Automated Tradespace Exploration



Details for Point 8284

- SubComp1=G
- SubComp2=1
- Config1=1.3419447
- Config2=5560.4277
- Cost=1471.1447
- Power Consumption=66.739334
- Weight=3.1583757
- Latency=0.27573425
- Preference Value=0.0
- BrushedBy Value=0.0

VRML Files

Image Files

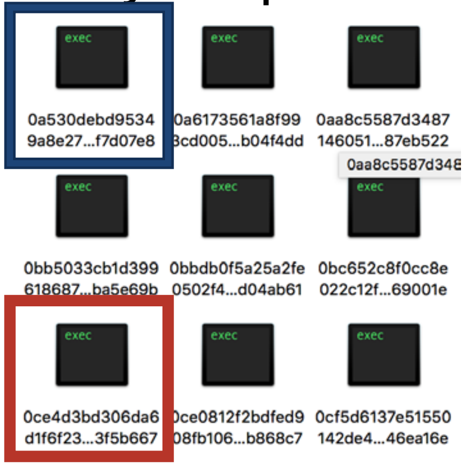
Other Files

Notes

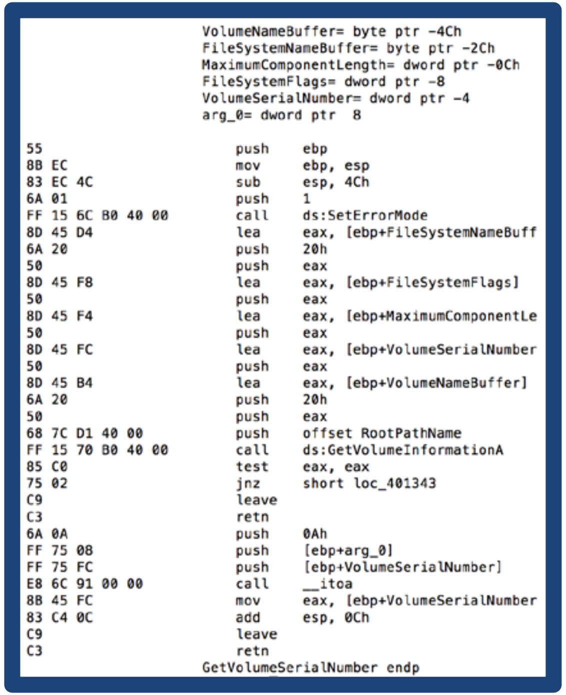
- Prototype connects AADL tooling to visual exploration software
- Any components that can be specified in AADL can be swapped in and analyzed
- Easily extended to include domain-specific analyses

Machine Learning for the DoD: Malware

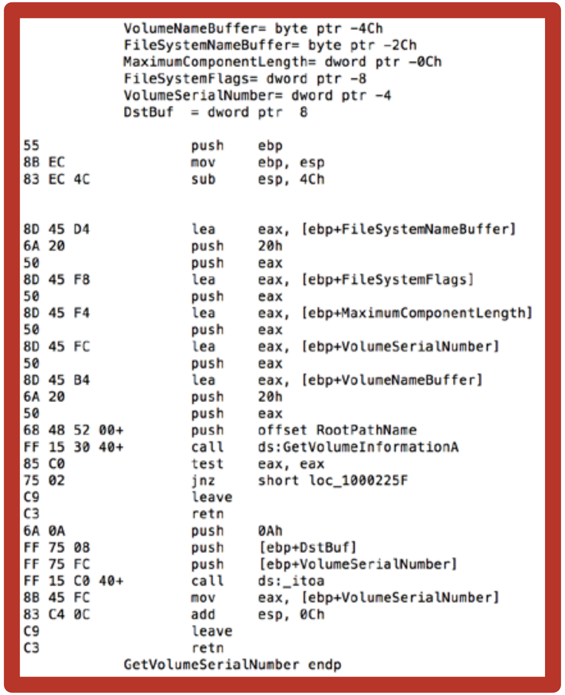
Many suspect files.



Manual pairwise analysis is expensive.



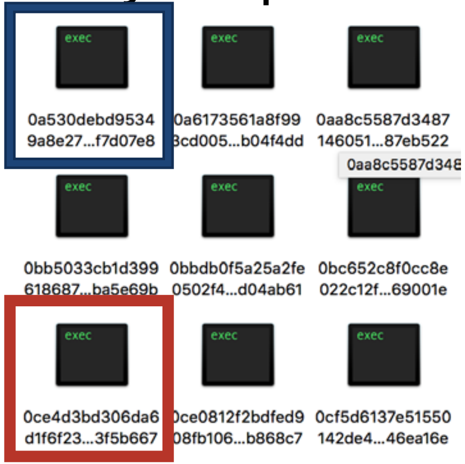
```
                                   VolumeNameBuffer= byte ptr -4Ch
                                   FileSystemNameBuffer= byte ptr -2Ch
                                   MaximumComponentLength= dword ptr -0Ch
                                   FileSystemFlags= dword ptr -8
                                   VolumeSerialNumber= dword ptr -4
                                   arg_0= dword ptr  8

55                                      push    ebp
8B EC                                   mov     ebp, esp
83 EC 4C                                sub     esp, 4Ch
6A 01                                   push    1
FF 15 6C B0 40 00                       call    ds:SetErrorMode
8D 45 D4                                lea     eax, [ebp+FileSystemNameBuff
6A 20                                   push    20h
50                                      push    eax
8D 45 F8                                lea     eax, [ebp+FileSystemFlags]
50                                      push    eax
8D 45 F4                                lea     eax, [ebp+MaximumComponentLe
50                                      push    eax
8D 45 FC                                lea     eax, [ebp+VolumeSerialNumber
50                                      push    eax
8D 45 B4                                lea     eax, [ebp+VolumeNameBuffer]
6A 20                                   push    20h
50                                      push    eax
68 7C D1 40 00                          push    offset RootPathName
FF 15 70 B0 40 00                       call    ds:GetVolumeInformationA
85 C0                                   test    eax, eax
75 02                                   jnz     short loc_401343
C9                                      leave
C3                                      retn
6A 0A                                   push    0Ah
FF 75 08                                push    [ebp+arg_0]
FF 75 FC                                push    [ebp+VolumeSerialNumber]
E8 6C 91 00 00                          call    __itoa
8B 45 FC                                mov     eax, [ebp+VolumeSerialNumber
83 C4 0C                                add     esp, 0Ch
C9                                      leave
C3                                      retn
                                   GetVolumeSerialNumber endp
```



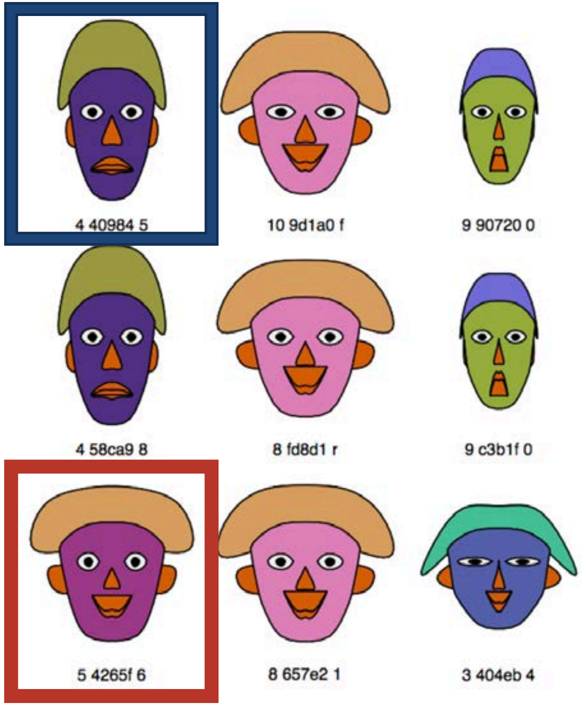
```
                    VolumeNameBuffer= byte ptr -4Ch
                    FileSystemNameBuffer= byte ptr -2Ch
                    MaximumComponentLength= dword ptr -0Ch
                    FileSystemFlags= dword ptr -8
                    VolumeSerialNumber= dword ptr -4
                    DstBuf  = dword ptr  8

55                         push    ebp
8B EC                      mov     ebp, esp
83 EC 4C                   sub     esp, 4Ch


8D 45 D4                   lea     eax, [ebp+FileSystemNameBuffer]
6A 20                      push    20h
50                         push    eax
8D 45 F8                   lea     eax, [ebp+FileSystemFlags]
50                         push    eax
8D 45 F4                   lea     eax, [ebp+MaximumComponentLength]
50                         push    eax
8D 45 FC                   lea     eax, [ebp+VolumeSerialNumber]
50                         push    eax
8D 45 B4                   lea     eax, [ebp+VolumeNameBuffer]
6A 20                      push    20h
50                         push    eax
68 48 52 00+               push    offset RootPathName
FF 15 30 40+               call    ds:GetVolumeInformationA
85 C0                      test    eax, eax
75 02                      jnz     short loc_1000225F
C9                         leave
C3                         retn
6A 0A                      push    0Ah
FF 75 08                   push    [ebp+DstBuf]
FF 75 FC                   push    [ebp+VolumeSerialNumber]
FF 15 C0 40+               call    ds:_itoa
8B 45 FC                   mov     eax, [ebp+VolumeSerialNumber]
83 C4 0C                   add     esp, 0Ch
C9                         leave
C3                         retn
                    GetVolumeSerialNumber endp
```

Machine Learning for the DoD: Malware

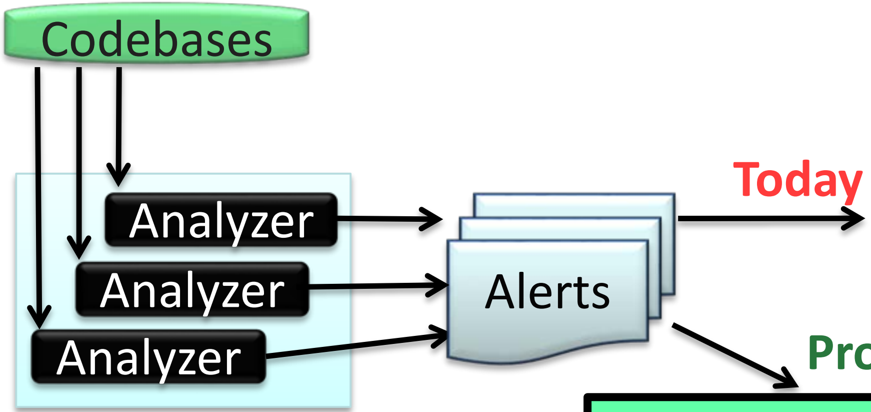
Many suspect files.

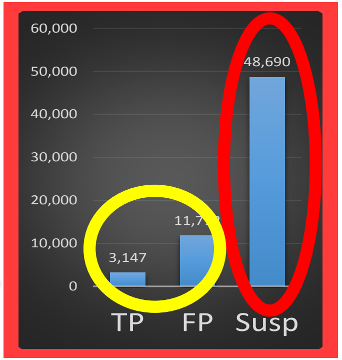

exec
0a530debd9534 9a8e27...f7d07e8
exec
0a6173561a8f99 3cd005...b04f4dd
exec
0aa8c5587d3487 146051...87eb522
0aa8c5587d348
exec
0bb5033cb1d399 618687...ba5e69b
exec
0bbdb0f5a25a2fe 0502f4...d04ab61
exec
0bc652c8f0cc8e 022c12f...69001e
exec
0ce4d3bd306da6 d1f6f23...3f5b667
exec
0ce0812f2bdfed9 08fb106...b868c7
exec
0cf5d6137e51550 142de4...46ea16e

Statistical visualization lowers costs.



Automated Analysis - Prioritizing Vulnerabilities





Many alerts left unaudited!

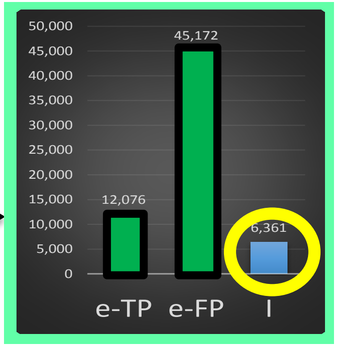
Project Goal

Classification algorithm development using CERT- and collaborator-audited data, that **accurately classifies most of the diagnostics** as:

Expected True Positive (e-TP) or

Expected False Positive (e-FP), and

the rest as Indeterminate (I)



Prioritized, small number of alerts for manual audit

<u>Long-term goal</u>: Automated and accurate statistical classifier, intended to efficiently use analyst effort and to remove code flaws

Image of woman and laptop from http://www.publicdomainpictures.net/view-image.php?image=47526&picture=woman-and-laptop "Woman And Laptop"

Automated Code Repair

Many violations of rules follow a small number of anti-patterns with corresponding patterns for repair

These can be feasibly recognized by static analysis

- printf(attacker_string) → printf("%s", attacker_string)

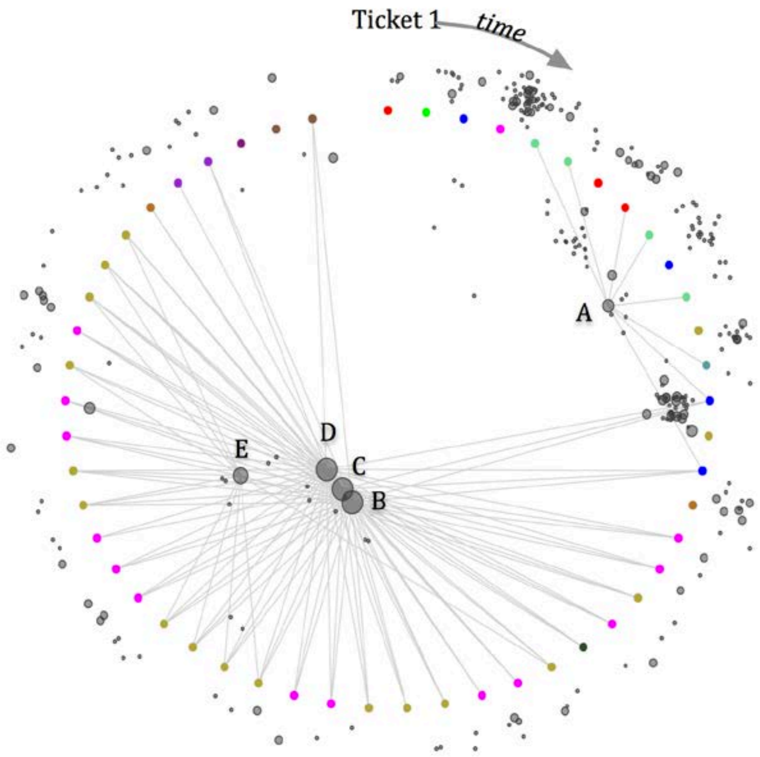
Creating tools to automatically repair these types of defects in source code

- Integer Overflows that lead to memory corruption
- Inferred memory bounds for reading from reused buffers
- Verified memory safety

Constraints

- The patched and unpatched program behave identically over the set of all traces that conform to the rules. (formally proven)
- No trace violates the rules. (formally proven)
- Repair in way that is plausibly acceptable to the developer.

Data Analysis for the DoD: Information Extraction



Cyber incident tickets are comprised of semi-structured data containing indicators

Traditional indicators like IP address, filename, file hash, email address can be augmented with concepts & relations

Presenters

Robert Schiela
rschiela@cert.org

Sam Procter
sprocter@sei.cmu.edu

Lena Pons
lepons@cert.org

Nathan VanHoudnos
nmvanhoudnos@cert.org

Jeff Boleng
jlboleng@sei.cmu.edu