Hardening Development Toolchains against Emergent Execution Engines (HARDEN)

Dr. Sergey Bratus
Program Manager
Information Innovation Office (I2O)

Proposers Day

September 30, 2021

Program objective

Develop practical tools to anticipate, isolate and mitigate emergent behaviors throughout the software lifecycle, to improve security outcomes in software for complex integrated systems

Controlling emergent behaviors is hard

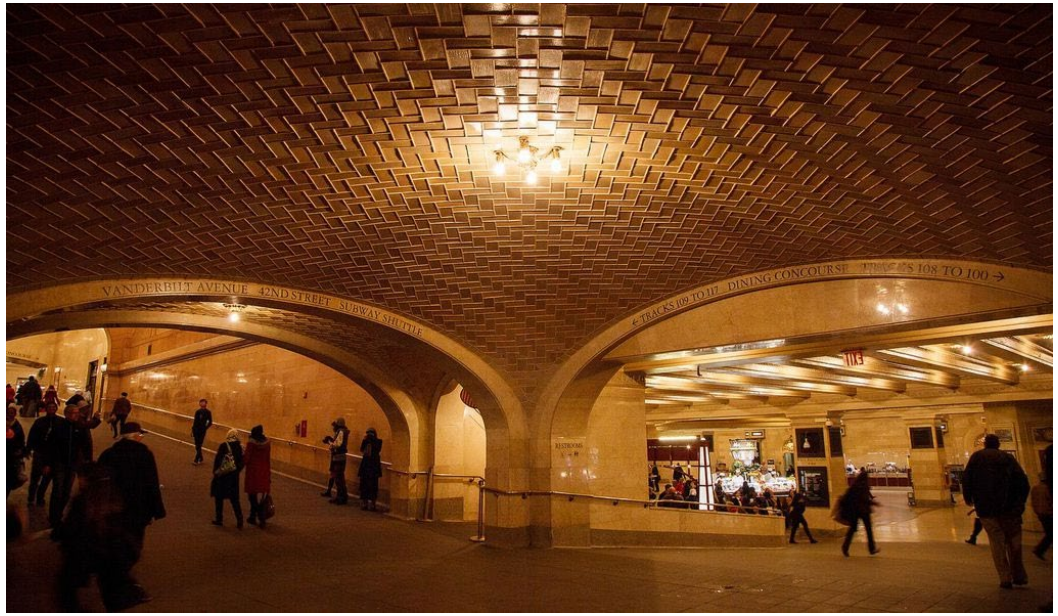
Design pattern

Parabolic ceiling of the U.S. Capitol Statuary Hall

Source: aoc.gov

Portability of exploit

Grand Central Terminal in Midtown Manhattan



Credit: David Hsu -Flickr/Creative Commons

Weird machine exploit: Overhear conversations

- Legend has it that John Quincy Adams positioned his desk at the focal point of the ceiling to overhear conversations of the opposition

Same design pattern: Parabolic ceiling

- Two people can stand on opposite sides of the arch and facing away from each other hold a conversation despite being 30 feet apart

It’s not the bricklayer's fault, but the architect’s design

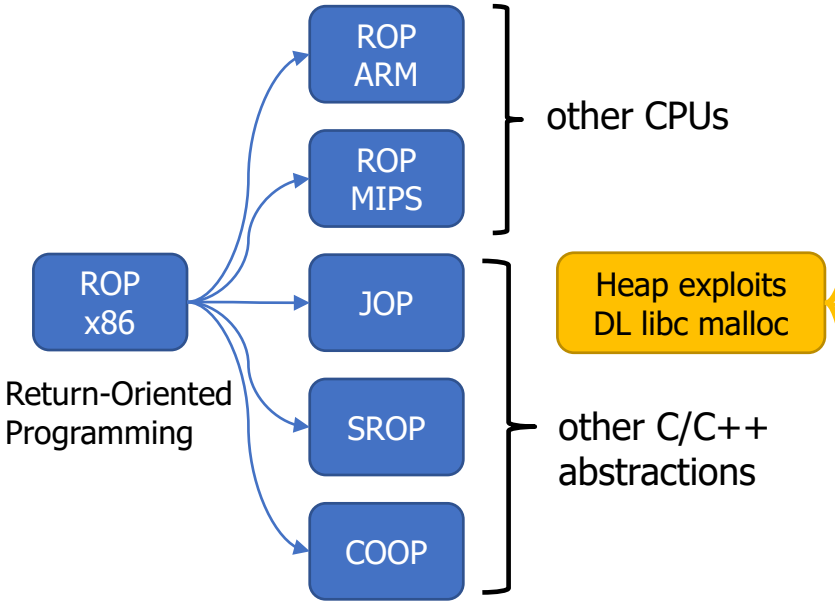
Challenge: Modern exploitation techniques are portable and robust

- Exploitation techniques are **portable** between applications, operating systems, and CPUs
- Attacks are often based on **reliable design patterns** of emergent behaviors **not** on particular **flaws** of a code base
- Exploits reuse the target's own unprotected **abstractions** at multiple interacting levels and form **patterns** around them

Three historic examples:

Stack

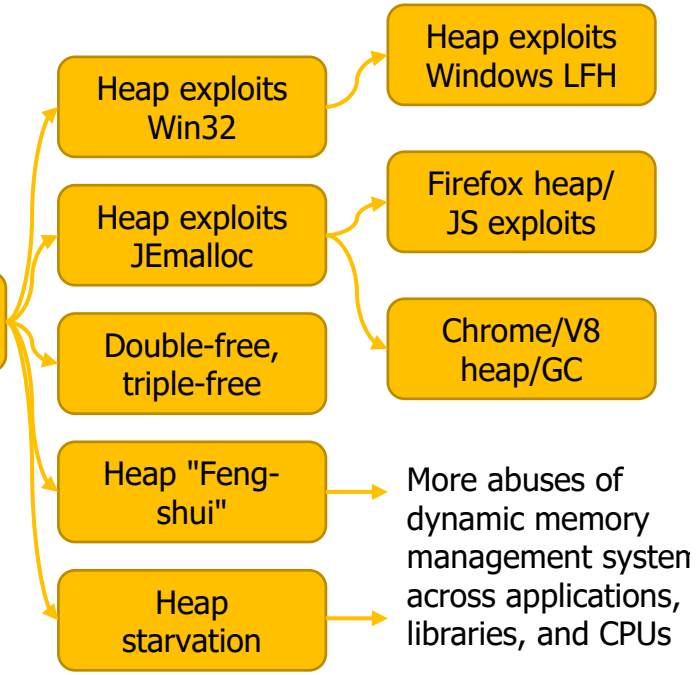
(late 1990s – mid 2000s)
Adversarial reuse of C/C++ control flow implementation patterns



JOP = Jump Oriented Programming
SROP = Signal Oriented Programming
COOP = Counterfeit Object Oriented Programming

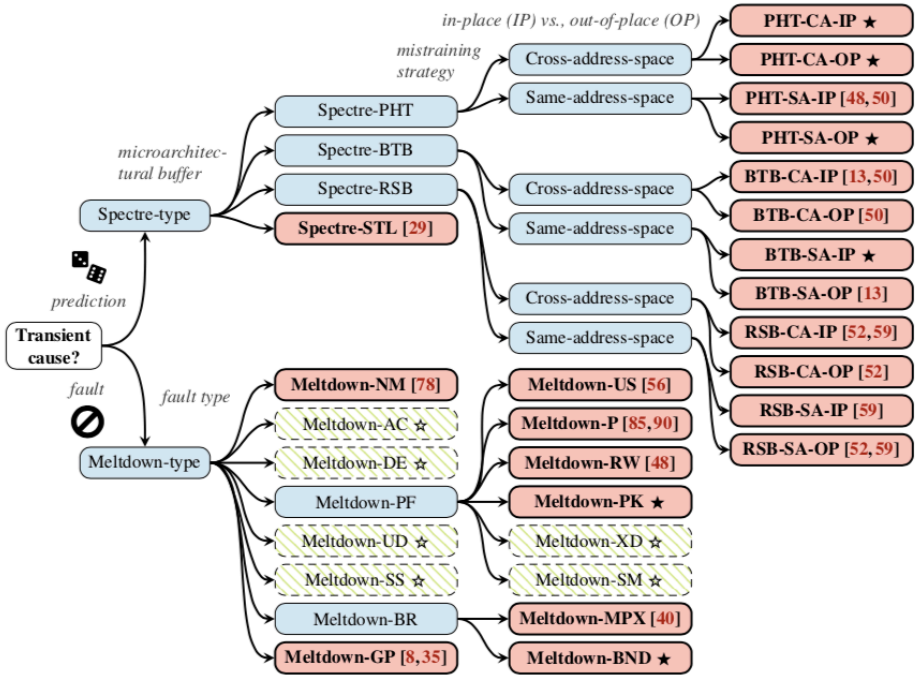
Heap

(early 2000s – mid 2010s)
Adversarial reuse of memory management patterns



CPU

(late 2010s – now)
Meltdown and Spectre types of vulnerabilities (currently 40+ and counting)

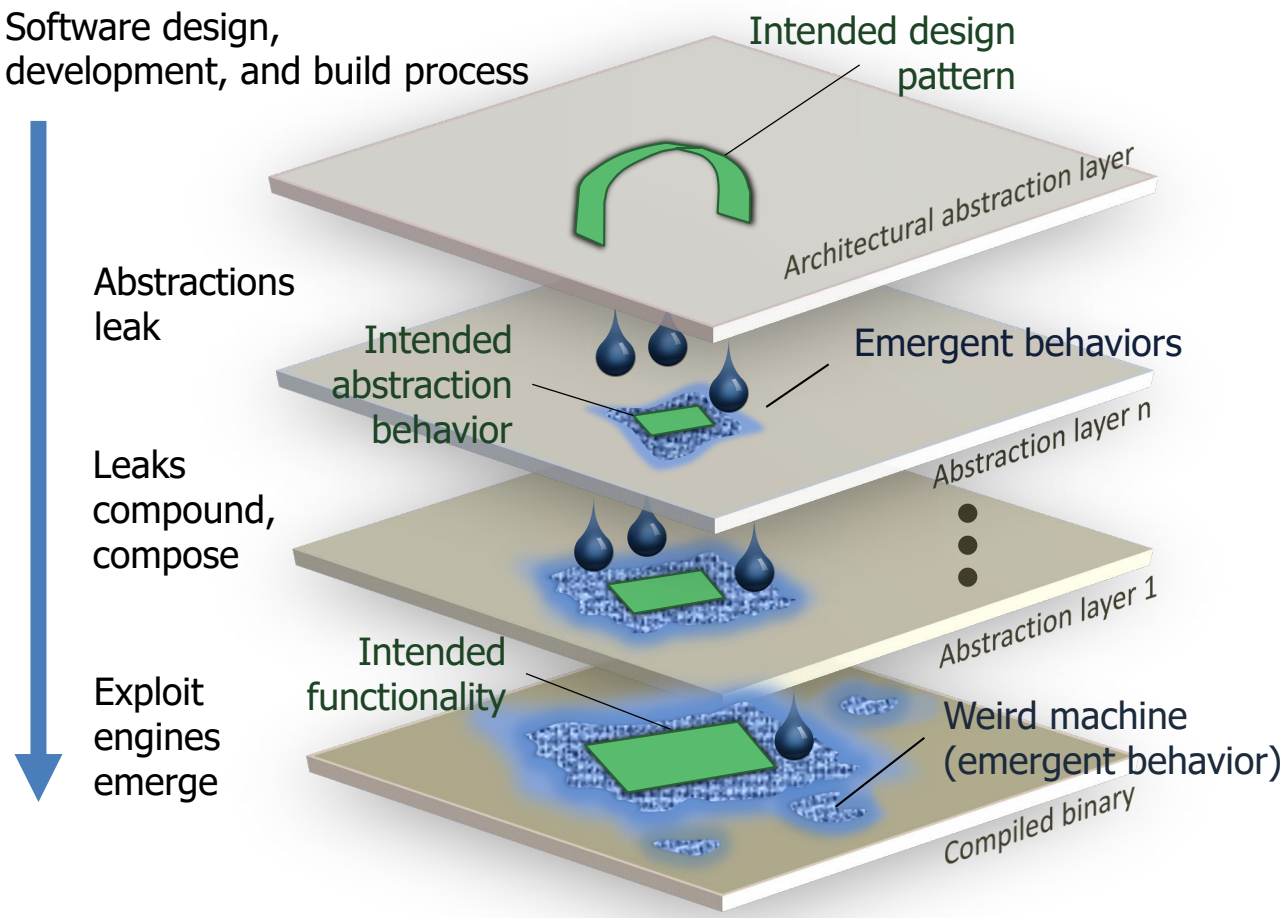


Challenge: Mitigations ignore design causes of emergent behaviors

- Mitigations are not applied at the right level of abstraction—fear of unexpected consequences, overfocus on implementation
- Initial locally focused mitigations are typically ineffective: small tweaks of exploits defeat them
  - Mitigations take several cycles to converge to effectiveness
- Advanced attackers are ahead of defenders, because their exploit harnesses model the target's exploitable abstractions

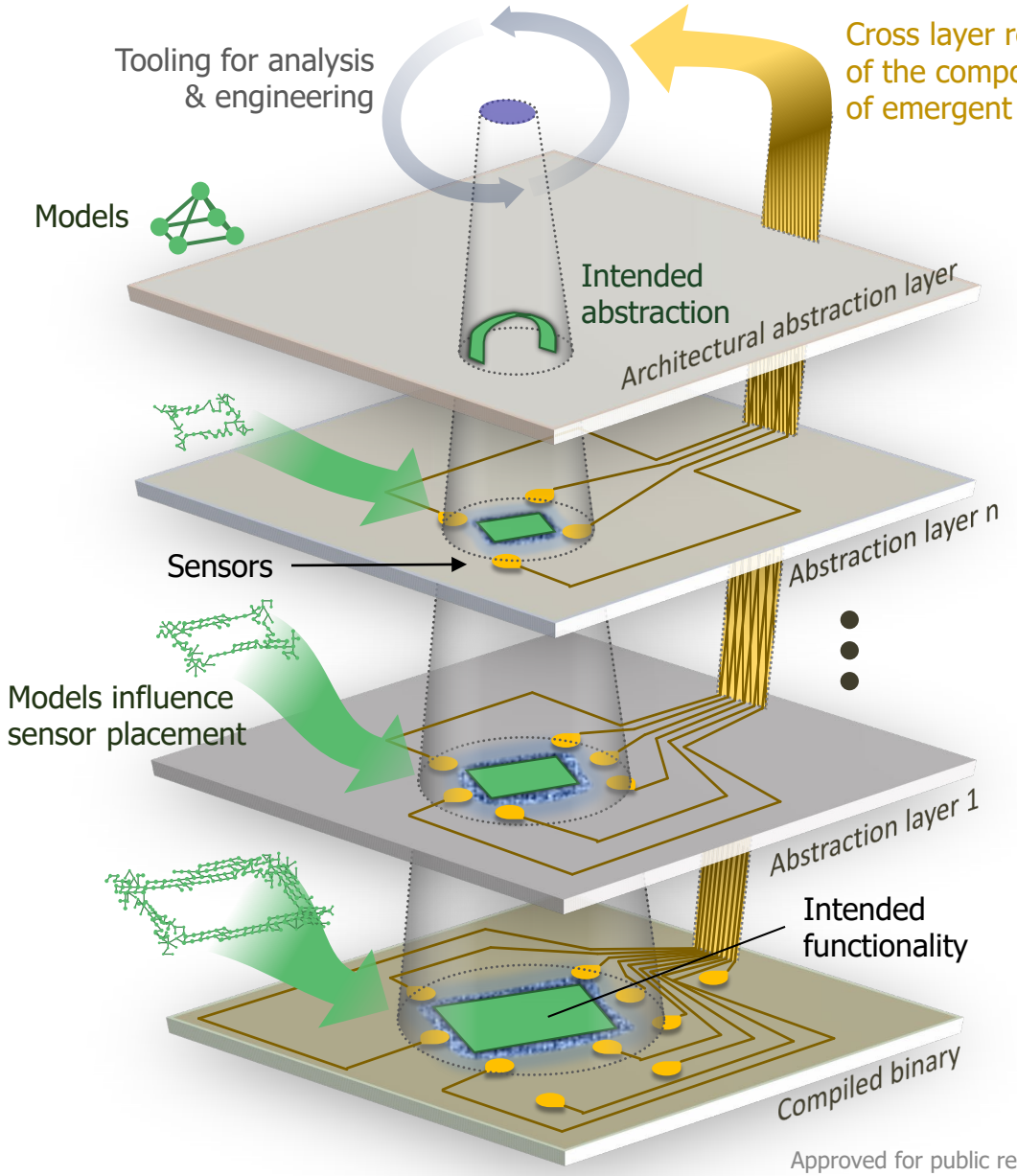
Root cause: Unintended composable emergent execution ("weird machines")

Today: Abstractions are unprotected;
no capability for reasoning about emergent behaviors



- Abstractions are implemented as *compositions* of lower-layer "building block" abstractions
- Lower-layer abstractions are designed to be composable; this enables *unintended* compositions (a.k.a., "abstractions are leaky")
- The fact that abstractions are both *unprotected* and *composable* leads to emergent behaviors
  - Code level: C, C++, JavaScript
  - Data level: crypto package signing, object formats
  - Design/algorithm level: heap management, caching
- Emergent behaviors accumulate into complex composable pools, reliable but unrelated to intended abstraction, enable *adversarial reprogramming*

Vision: Model and control emergent execution



- Intended abstractions are protected
- Accumulation of emergent behaviors is controlled
- Emergent execution engines ("weird machines") are suppressed

Approaches to protecting abstractions:

- Models of intended behavior
- Static reasoning to bound unintended composability
- Hardened runtime instrumentation against deviations
- Integration with design and development tools

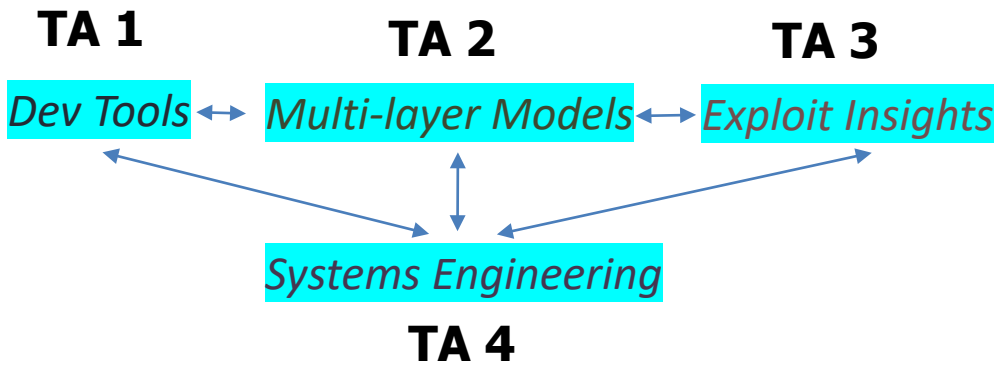
Technical areas

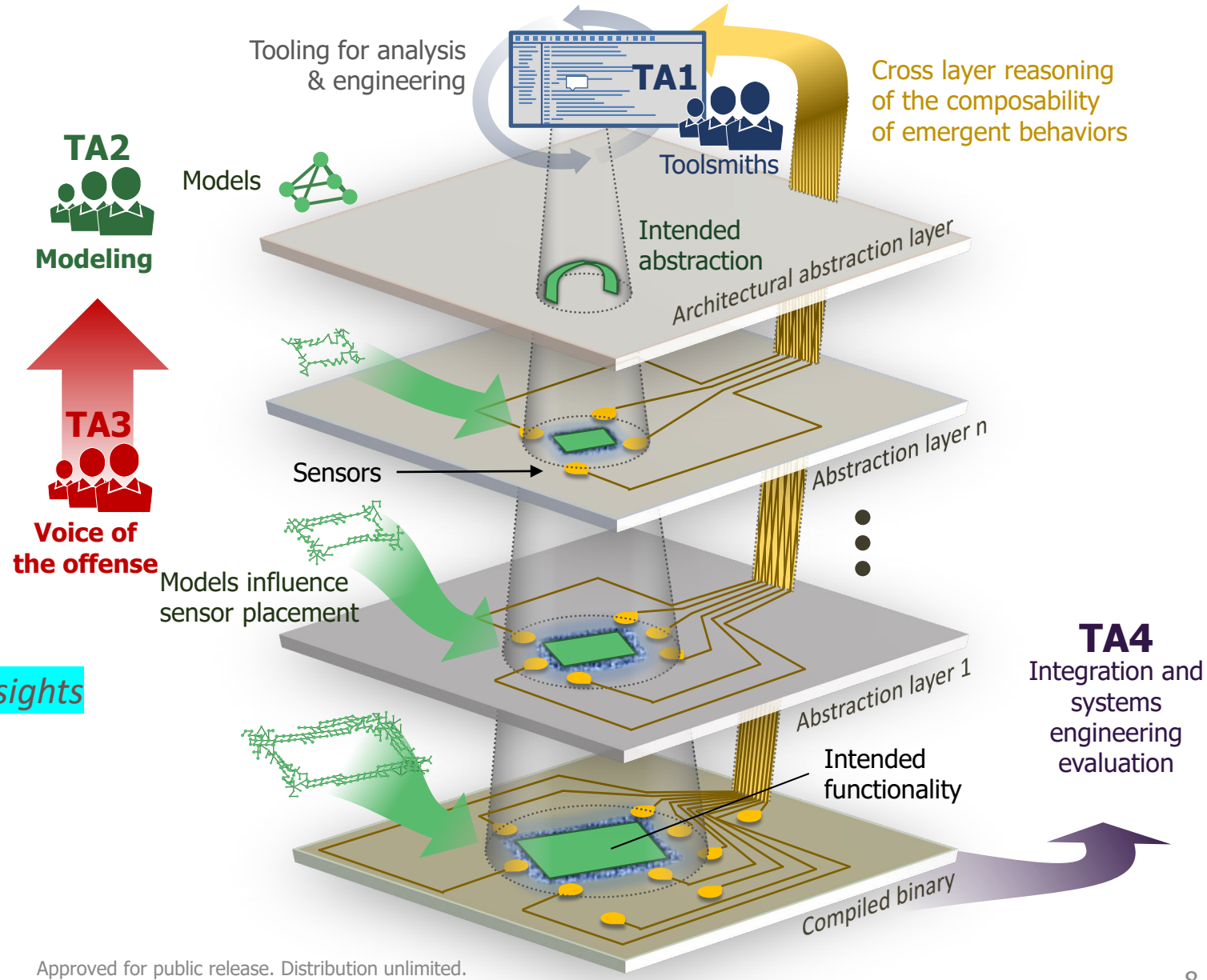
TA 1: Tooling for developers

TA 2: Modeling of emergent behaviors

TA 3: Voice of the offense

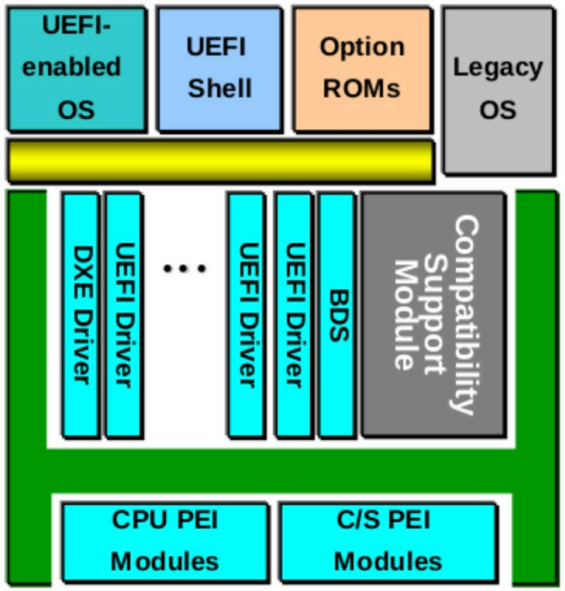
TA 4: Integration and systems engineering evaluation





BAA's exemplary technological use cases for evaluation

UEFI, Chain-of-Trust



Hardened sensor system based on UEFI/SOSA standards for chain-of-trust (trusted sensor)

Pilots' tablet integration

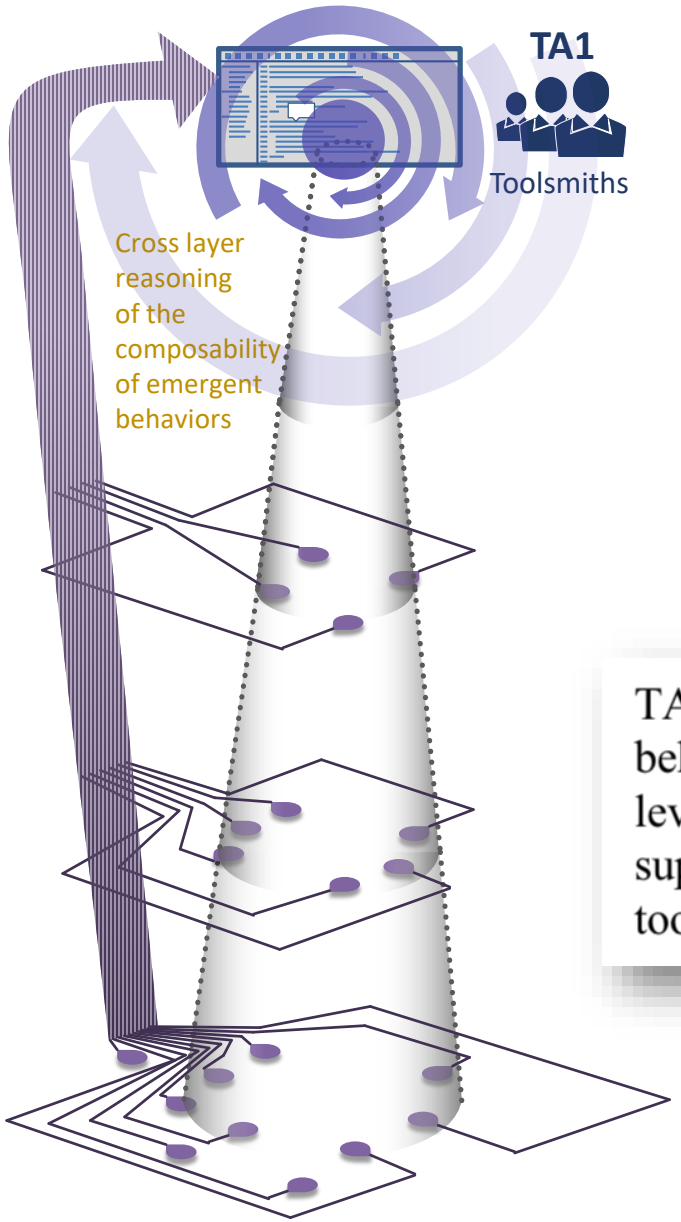
Source: airforcemag.com

Hardened integration technological stack for a COTS-based pilot's tablet to interface with aircraft's trusted mission computer

> Although HARDEN seeks to create broad theories and generic tools, the program will focus on validating its approaches by applying them to concrete technological use cases of integrated software systems described further under "Technological Use Cases" within Section I.B.

SOSA: Sensor Open Systems Architecture

TA 1: Tooling for developers

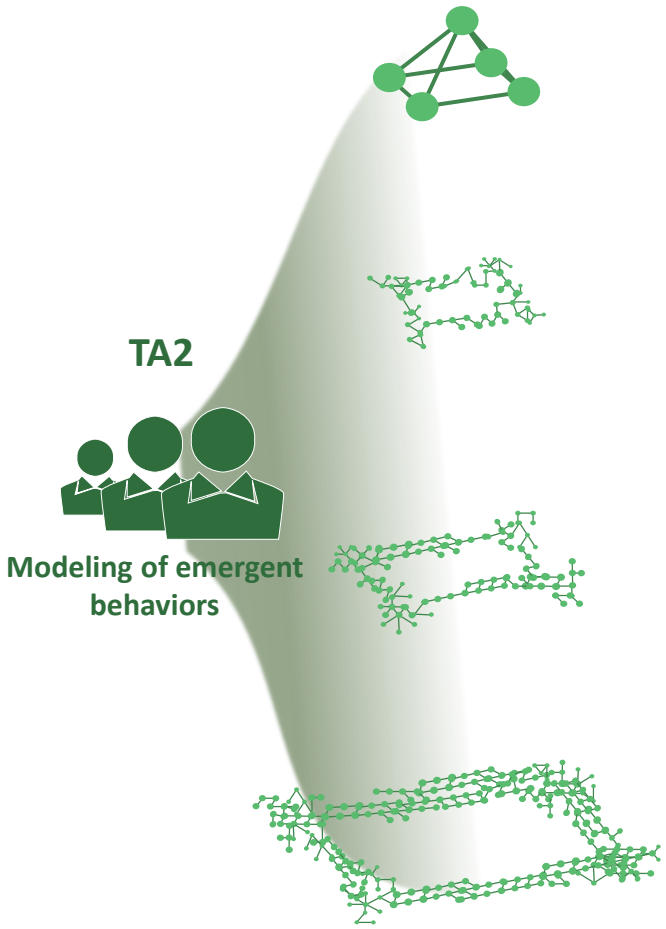


Challenges

- Overcome state explosion of typical models of software behavior
- Make annotation of expected behavior and predictions of emergent behavior accessible to regular software developers
- Develop efficient means of communicating about Emergent Execution (EE) with developers
- Integrate anticipation of EE with common developer workflows and tools

TA1 performers will develop and combine novel approaches for scalable reasoning about behaviors of computing systems' units, layers, components, and subsystems, to support multi-level modeling of system state evolution and emergent behaviors of unprotected abstractions. To support such reasoning, TA1 will develop novel theories, models, metadata, instrumentation, and tools.

TA 2: Modeling of Emergent Execution (EE) behaviors

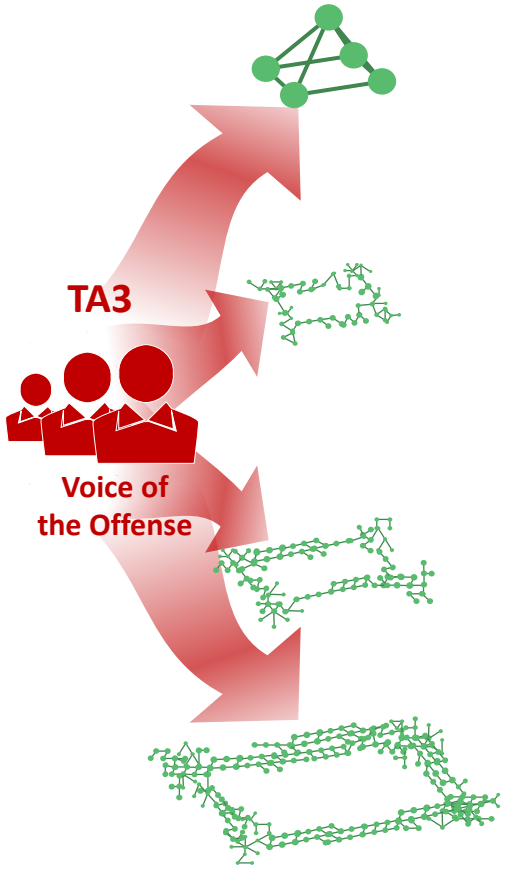


Challenges

- Create **models of emergent execution** that capture designed-in EE and abstract away irrelevant parts of the implementation
- Model interfaces and APIs at several layers of abstraction, and their interactions
- Develop effective tiered **representations** of abstractions to reason about EE, and formats to efficiently store and retrieve these representations alongside software deliverables (cf. debugging symbolic data formats)

TA2 performers will focus on creating the capability for platform experts to model EE behavior in the use cases described under "Technological Use Cases" within Section I.B. TA2 performers will develop modeling methods, languages, and tools for characterizing emergent execution at different abstraction levels; produce and represent tiered and scalable models suitable for fast TA1 feedback; and will create models of EE for all relevant interfaces in the integrated system use cases.

TA 3: Voice of the offense

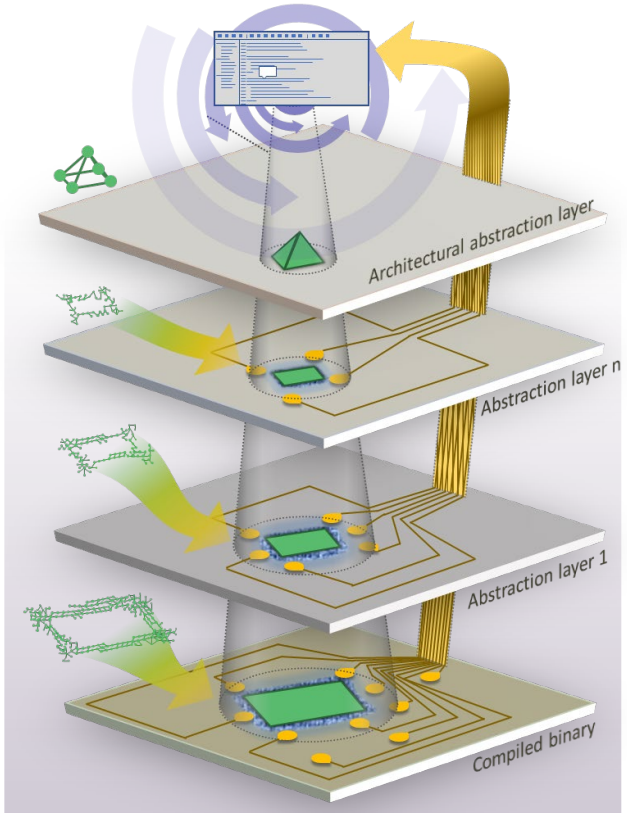


Challenges

- Gather and generalize dispersed exploitation expertise across technical domains, representative of the edge-of-the-art
- Translate exploit development intuition and tradecraft for the formal modeling approaches of **TA2**
- Test actual effectiveness of proposed EE mitigations

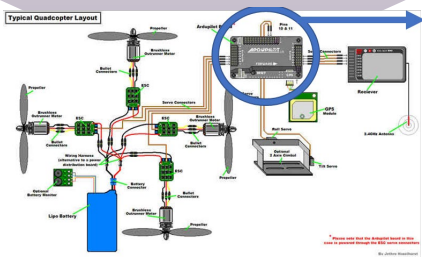
The TA3 performer will focus on the generalization of edge-of-the-art exploitation patterns in close coordination with TA2 performers to help model EE and exploitation. The TA3 performer will identify and describe integrated system understanding and exploration techniques used in public exploitation of complex targets to locate composable primitives for multi-step exploitation methods.

TA 4: Integration and systems engineering evaluation



TA4

Integration and systems engineering evaluation



Challenges

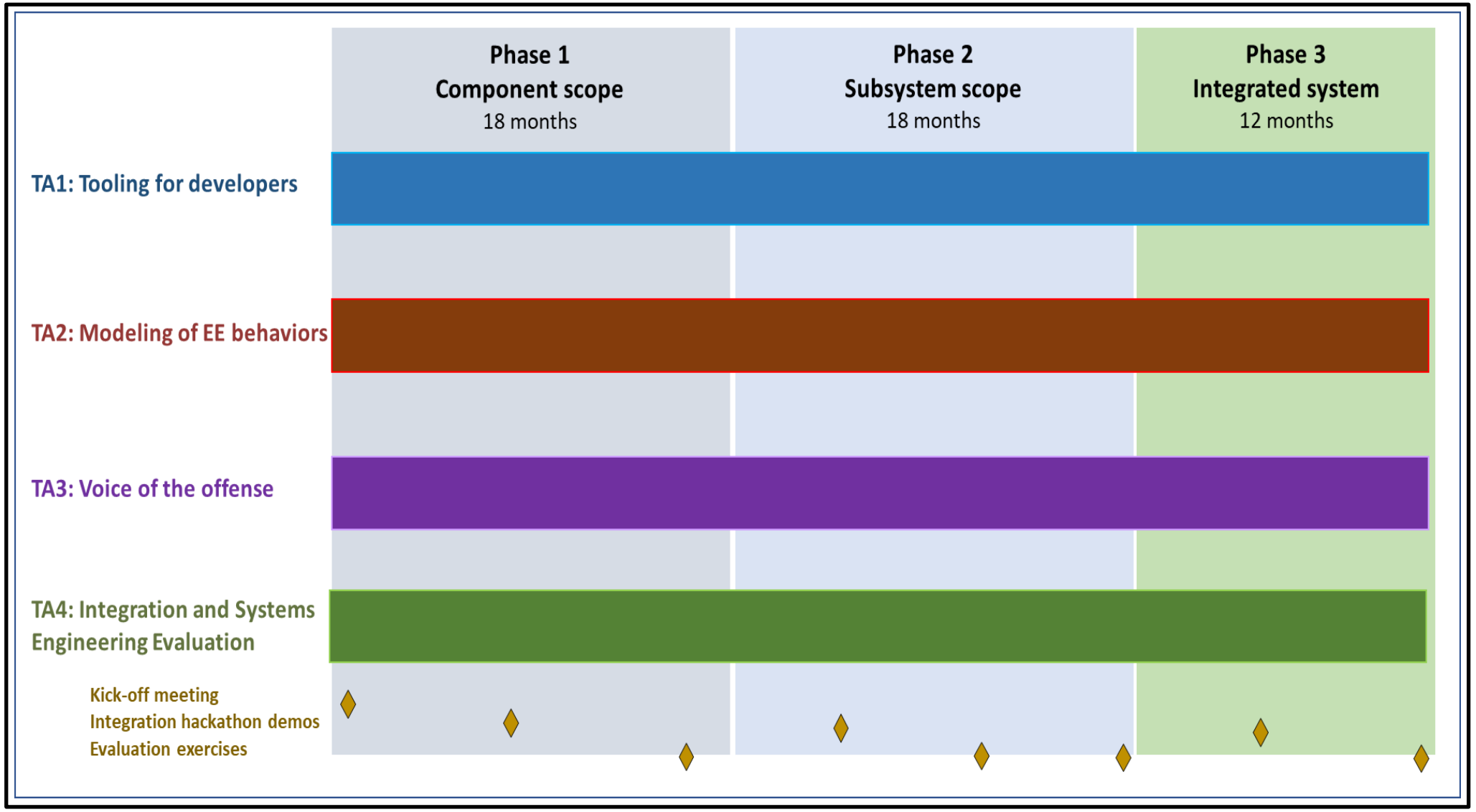
- Deploy tools and models developed by **TA1** and **TA2** to anticipate and mitigate EE in notable open-source integrated software systems relevant to BAA's technological use cases and suitable for fundamental research
- Produce the testbed to demonstrate **TA1**, **TA2**, and **TA3** technological capabilities, and to evaluate them against program metrics in coordination with the **TA3** performer
- Set up and manage transition to DoD systems of interest

The TA4 performer will provide system integration and evaluation, applying tools developed by TA1 performers, and models developed by TA2 performers, to demonstrate reliable and effective hardening of a sensor system based on UEFI chain-of-trust ("trusted sensor") and of pilots' tablets and trusted mission system integration layer ("Cockpit tablet/User Interface (UI)").

Evaluation metrics and milestones

	Metric	Phase I (18 months) Component scope	Phase II (18 months) Subsystem scope	Phase III (12 months) Integrated system
All	Lines of Code, C/C++	50—100K	800K—1M	10—20M
	Exemplary software complexity	OpenWRT core IoT router/bridge firmware	TianoCore EDK2 UEFI firmware	Android (AOSP) subset/tablet ROS2/DDS avionics firmware (UAV)
TA1 & TA2	Instrumentation overhead	<=15%	<=10%	<=5%
	Time to transformation accuracy	1-2 months	<=4 weeks	<=4 days per component
	Coverage of objects and interfaces	60%, manual selection of test surface	80%, automated, with human-in-the-loop	95%, fully automatic test surface selection
	Alert / mitigation effectiveness	>=70% of tested emergent behaviors mitigated	>=80%	>=90%
TA3	Analysis efficiency over SoTA red team	10x on average	Up to 100x, 30—50x on average	1000x

Program schedule



Meetings and reporting requirements

- **Two annual** Principal Investigator (PI) meetings centered around a challenge problem
- PM site visits between PI meetings
- Annotated slide presentations will be submitted within two weeks after program kick-off meeting and after each review
- **Quarterly technical** progress reporting
  - Technical report describing progress, resources expended and issues requiring Government attention, provided 10 days after the end of each quarter
- **Monthly financial** reporting
- Financial/technical progress reporting to DARPA's Deliverable Repository (VAULT)
- System Development Plan provided one month after the kick-off meeting for each phase
  - Describe the scope/design and hardware and software architecture
- Software and software Documentation – All computer software delivered under the HARDEN program must be delivered as source and object executable code. Include the source listings and source code for the target computer systems, as well as any build scripts or other technical information required for DARPA to compile all delivered source code.
- Hardware designs and documentation
- Technical data generated by the program
- Phased and Final Technical Report

Funding and programmatic details

- **Proposals due: Tuesday, November 4, 2021 at 12:00 noon (ET)**
- Government anticipates multiple awards for TA1 and TA2, and a single award for TA3 and TA4
  - Procurement contracts, Cooperative Agreements, or Other Transactions (OT)
- Proposers may submit multiple proposals
  - Each proposal may address any one TA, or a combination of TA1 and TA2
    - If submitting a combination of TA1 and TA2 each TA must be **clearly distinct and separable** by costs and Statement of Work
    - A proposer submitting proposals to TA1 and TA2 may be selected to perform on one or both of these TAs
    - TA3 and TA4 proposers cannot be selected to perform on any other TAs
    - Which to consider for award (if any) is at the discretion of the Government
  - To expedite award contracting, proposers are encouraged to have sub-award agreements in place ahead of award notification

Funding and Programmatic Details

BAA Location

- Posted on the **Contract Opportunities** website (https://sam.gov/opp/3f6ee0a93e4844e59e3681e2cdf05936/view) and **Grants.gov** website (https://www.grants.gov/web/grants/view-opportunity.html?oppId=335831)

Questions Today

- Questions can be submitted until 1:35 PM ET via HARDEN@darpa.mil. Please do not post questions in Zoom.
- Questions not answered verbally during today's Q&A session will be addressed through the FAQ. This will get regularly updated and posed on https://www.darpa.mil/work-with-us/opportunities.

Information precedence

- If anything said or addressed during this presentation or in the FAQ conflicts with the published solicitation, **the BAA takes precedence**. The Government may issue amendments to the BAA to effect any changes deemed necessary in response to the FAQ. Such amendments would be posted to Contract Opportunities (https://sam.gov) and Grants.gov (https://www.grants.gov) prior to the solicitation closing date and would supersede previous versions of the solicitation.

www.darpa.mil