

CYBER SECURITY DIVISION 2013 PRINCIPAL INVESTIGATORS'

# Code Pulse: Dynamic Augmented Static Analysis

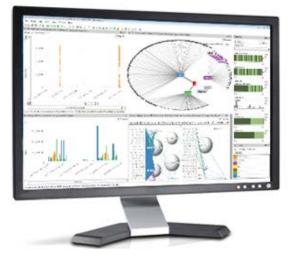
Secure Decisions Hassan Radwan

Sept. 16<sup>th</sup>, 2013



# **Secure Decisions**





#### We help you make sense of data

- Analyze security *decision-making* processes
- Build *visual analytics* to enhance security decisions and training

Our expertise starts where automated security sensors and scanners leave off

We transition our R&D into operational use, in government and industry



Grounded in commercial software and product development

- Division of Applied Visions, developer of commercial software
- 40 people, most with clearances, and secure facilities

## **Secure Decisions SwA Tools**

### Code Dx

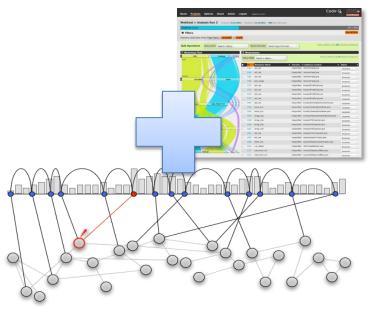
DHS funded Phase II SBIR Currently TRL 7

ome Projects	Options	About Admin	Logout			Code	
ebGoat > Ana	lysis Run 3	Constant on 6/14/1	012 104	maint on 6/14/2012	830 total weaknesses		
leakness count	.,						272 / 830
Filters							dear all filters
	hose Triage State	us is Excelated O or Fi	0 10				
Bulk Operations	Status Setter	Select a status		Report Generator	Select report format *	Actions will be for the 2	72 matching weaknesse
Weakness Flow		_		eaknesses			
PMD		FindBugs 3k				Actions will be for th	e O selected weakresse
loo		10	State	is Setter Select a st	atus *		
				Id • Weakness N	ame • Severity (	Codebase Location	Status
				1963 null_var	Unspecified	HammerHead.java	Escalated
	1 10			1564 null_var	Unspecified	HammerHead.java	Escalated
atus			0	1965 null_var	Unspecified	HammerHead.java	Escalated
Escalated		Fixed	0	1966 lock_assign	Unspecified	HammerHead.java	Escalated ·
			0	1571 null_var	Unspecified	lessons/BlindScript.java	Escalated
	THO I			1572 null_var	Unspecified	lessons/BlindScript.java	Escalated
				1573 nul_var	Unspecified	lessons/BlindScript.java	Escalated
				1574 null_var	Unspecified	lessons/BlindScript.Java	Escalated
Basic Rules	BAD P	RACTICE	0	1591 null_var	Unspecified	lessons/FailOpenAuthentication.java	Escalated
Sand Roles	040 1	NI NI	10	1592 weak_cmp	Unspecified	lessons/JavaScriptValidation.java	Escalated
				1593 weak_cmp	Unspecified	lessons/Java5criptValidation.java	Escalated
	1-1		0	1597 string_cmp	Unspecified	lessons/WeakAuthenticationCookie.java	Escalated
	D 110		0	1599 string_cmp	Unspecified	lessons/XMLInjection.java	Escalated
eakness name	K 1/10		0	1600 string_cmp	Unspecified	lessons/XMLInjection.java	Escalated
	other E	imptyC		1601 string_cmp	Unspecified	lessons/XMLInjection.java	Escalated
	1111			1602 null_var	Unspecified	lessons/XPATHInjection.java	Escalated
				1607 null_var	Unspecified	session/LessonTracker.java	Escalated
				1608 weak_cmp	Unspecified	session/RandomLessonTracker.java	Escalated
1.00				1614 run_nosync	Unspecified	util/ThreadWatcher.java	Escalated
everity.	Medium			1717 concurrent_cal	I Unspecified	session/DatabaseUtilities.java	Escalated
	Healum	Mine Mine		1718 concurrent cal	Linspecified	session/DatabaseUtilities.Java	Escalated

# Converged static source analysis

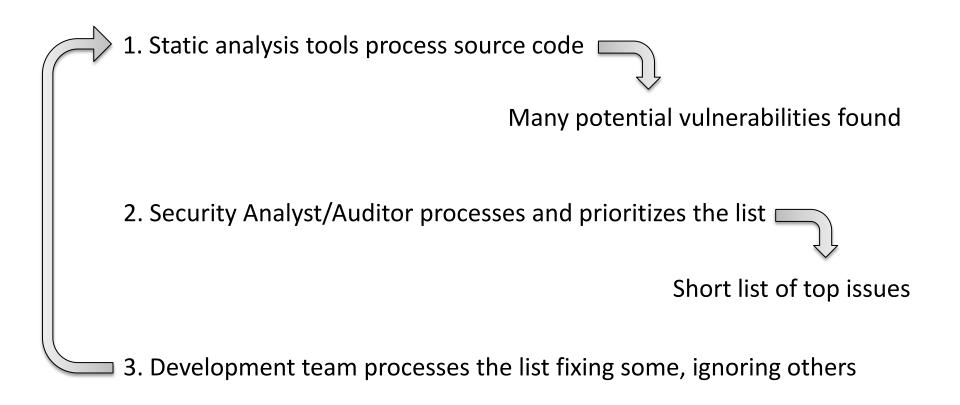
### **Code Pulse**

DHS CSD funded BAA 20 month project, PoP through April '14

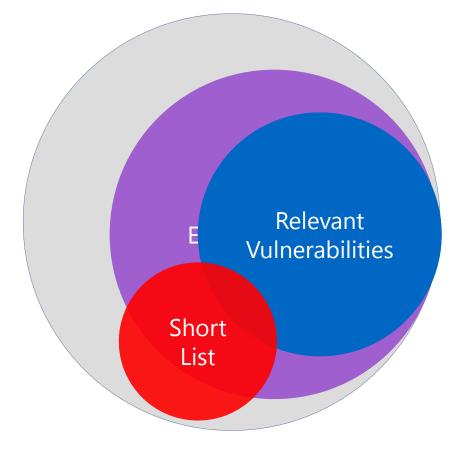


Dynamic augmented static source analysis

# **Typical Static Analysis Workflow**



## **Need: Relevant Vulnerabilities**



## **Need: Remediation Context**



#### Time

Developers fixing a bug/vulnerability when the source context is still fresh will be quicker and less error-prone

As time progresses their speed and ability to ensure system integrity with new changes diminishes

# Terminology

#### **Static Analysis**

- At the source or binary levels
- Scans to detect potential vulnerabilities
- No runtime context

#### **Dynamic Analysis**

- In SwA used to describe the process to detect potential vulnerabilities at runtime (A.K.A. Penetration Testing, Black Box Testing)
- Source code context not available

#### **Dynamic Tracing**

- Monitor runtime execution
- Used by profiling tools
- Identify which methods are called and when to observe: call graph, call durations, and call frequencies



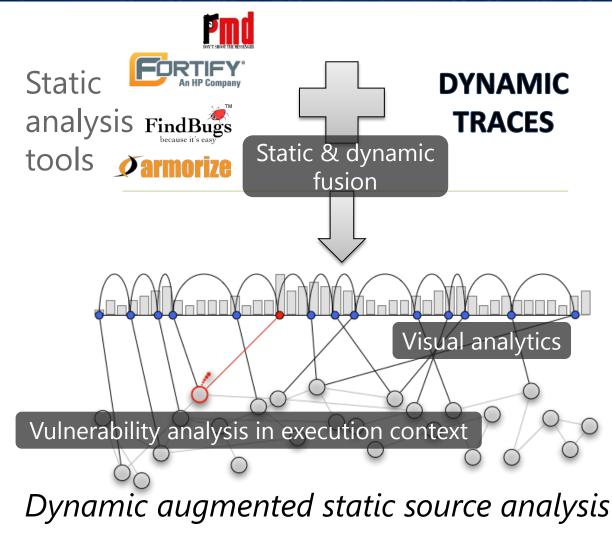
### **Static source analysis**



### **Dynamic tracing**

for Java





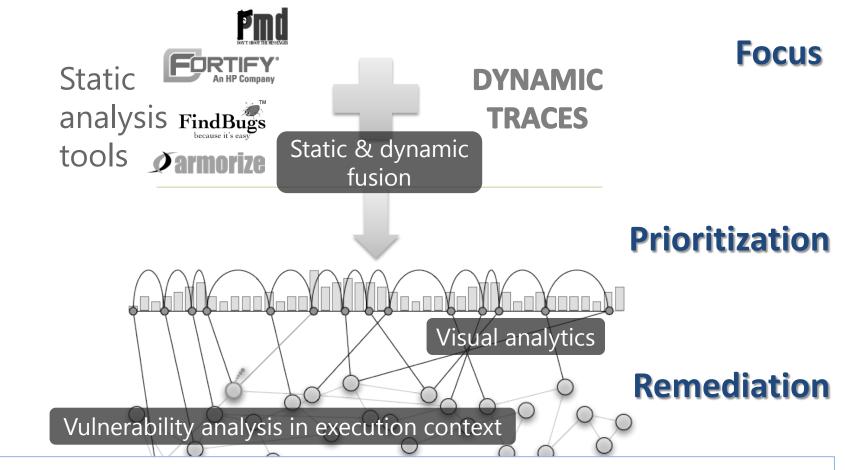


#### 3 primary high-level use cases





### **Benefits**

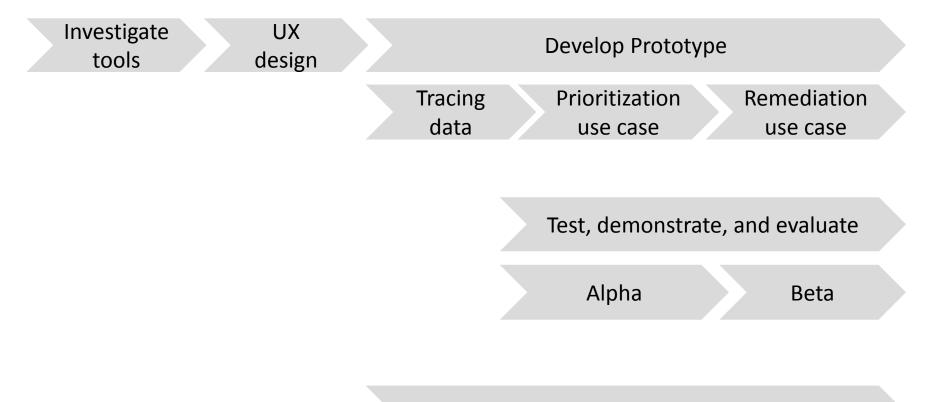


# Focus the prioritization and remediation of static software analysis

# Competition

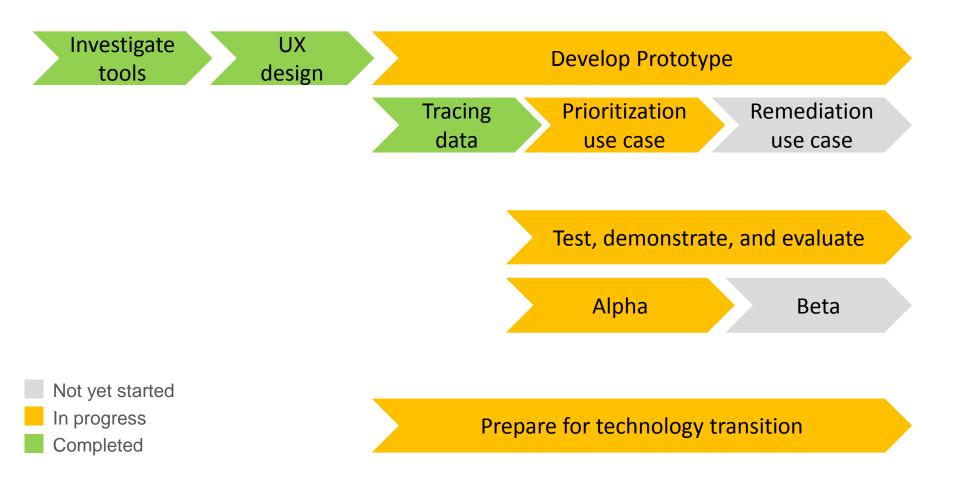
- Dynamic application security testing
  - Black box testing
  - Little insight into the internals of the target software
- Hybrid application security testing
  - Combination of static and dynamic testing
  - Better security coverage
  - But static analysis results still need to be processed using traditional techniques without runtime context
- No existing solution for automated runtime correlation with static analysis results

### **Task Overview**

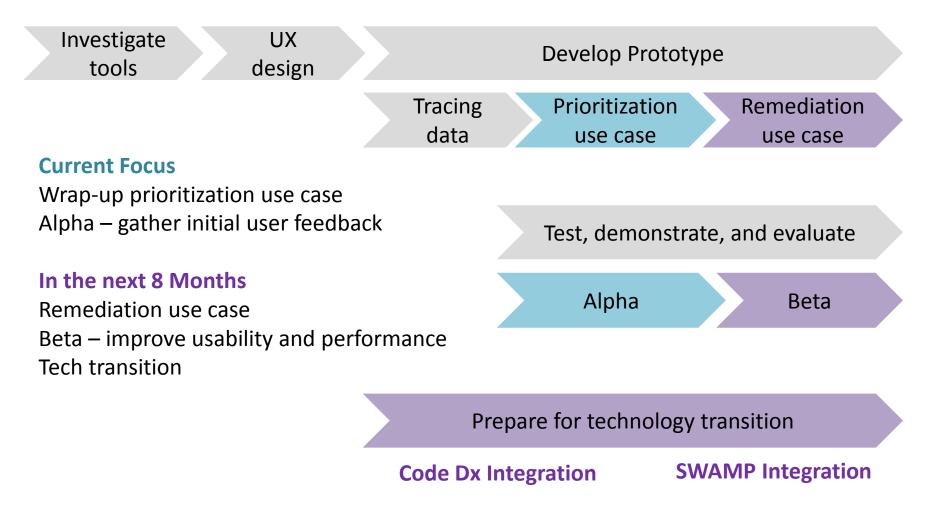


#### Prepare for technology transition

## **Current Status**







# **Contact Information**



#### Hassan Radwan

# hassan.radwan@securedecisions.com 518-207-3106