SWINBURNE UNIVERSITY OF TECHNOLOGY

Model-driven Software Security Engineering for the Cloud

Mohamed Almorsy,
Post-doctoral Fellow

Amani Ibrahim, PhD Student

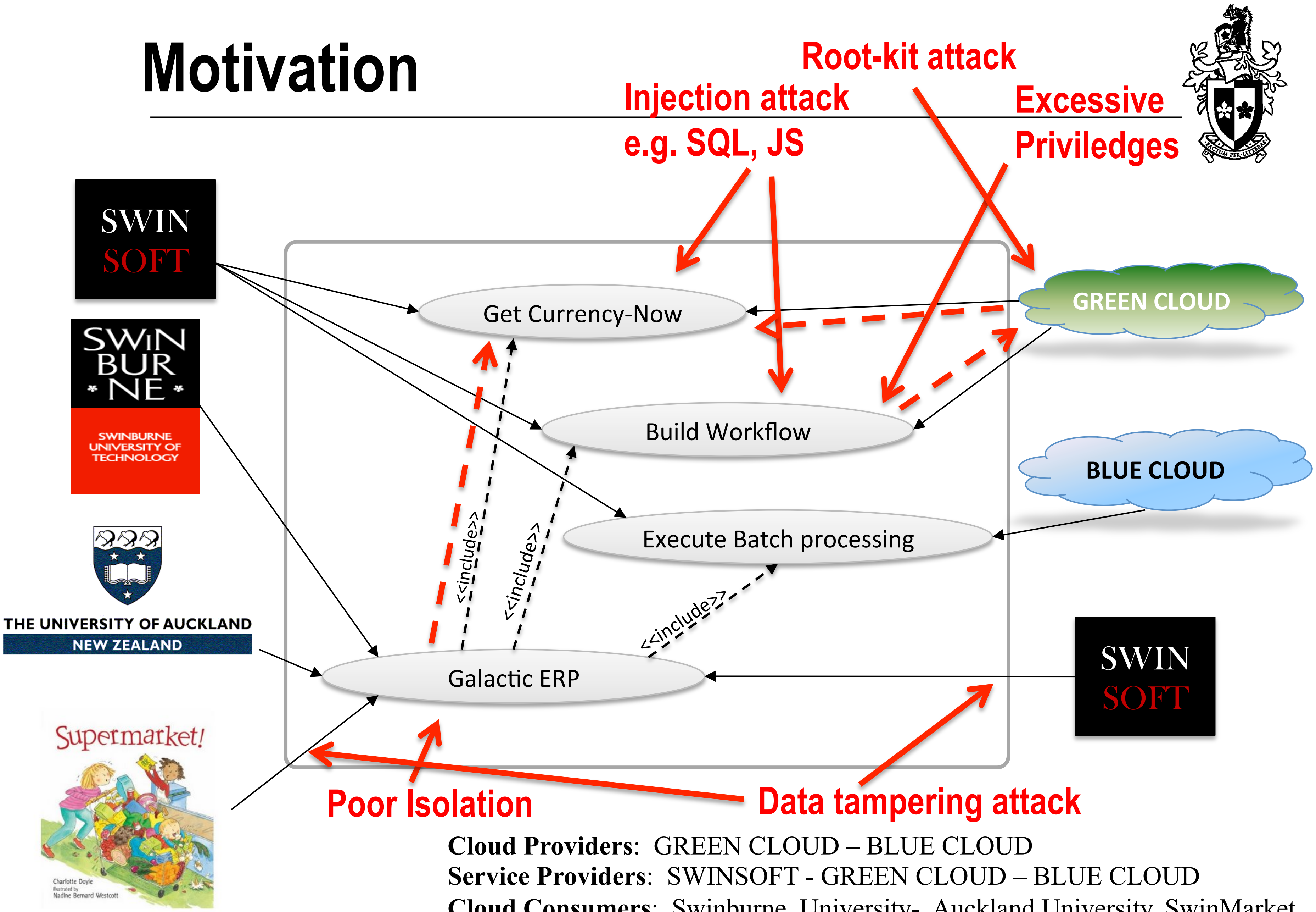
John Grundy, Deputy Dean, FICT

Outline

- Motivation
- Cloud computing 101
  - What the heck are IaaS, PaaS, SaaS anyway???
- CloudSec – protection @ IaaS level against root-kits
- MDSE@R – flexible security for PaaS/SaaS levels
- Future Directions

- Part 1:
  - Motivation
  - Overview of cloud computing concepts
  - Our approaches

Motivation



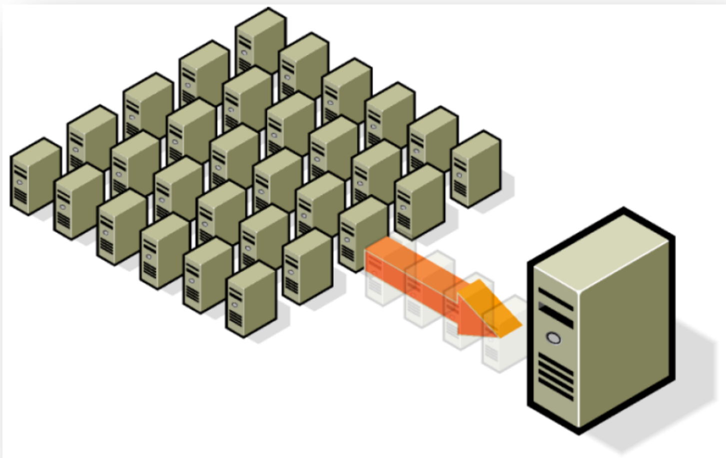
Cloud Providers: GREEN CLOUD – BLUE CLOUD

Service Providers: SWINSOFT - GREEN CLOUD – BLUE CLOUD

Cloud Consumers: Swinburne University- Auckland University, SwinMarket

Cloud Computing 101

- Resource virtualisation e.g. VMWare
- Elasticity, Pay-per-use vs buy & maintain
- Infrastructure as a Service (IaaS) e.g. Amazon
- Platform as as Service (PaaS) e.g. Google App Engine
- Software as a Service (SaaS) e.g. SalesForce.com
- Multi-tenant applications



Key Security Problems w Cloud Model

- IaaS:
  - Cloud providers don't know whats running on VMs
  - Cloud users don't know what other apps running / platform security policies
- PaaS:
  - Design-time focus of security solutions BUT security needs emerge @ run-time
  - Lack of integration of security / cloud app architecture
- SaaS:
  - Evolving tenant needs / limited (no?) tenants involvement in security configuration

Our Approach(es) to address…

- IaaS protection:
  - **CloudSec** – security appliance for hypervisor layer
  - Supported by points-to analysis tool (KDD) and kernel object discovery algorithm (DIGGER)
- PaaS:
  - **MDSE@R** – model-driven security engineering with run-time updating of deployed cloud applications
  - Supported by vulnerability analysis & mitigation, re-aspects
- SaaS:
  - TOSSMA – cloud consumer security management console
  - SMURF – multi-tenant re-engineering via re-aspects

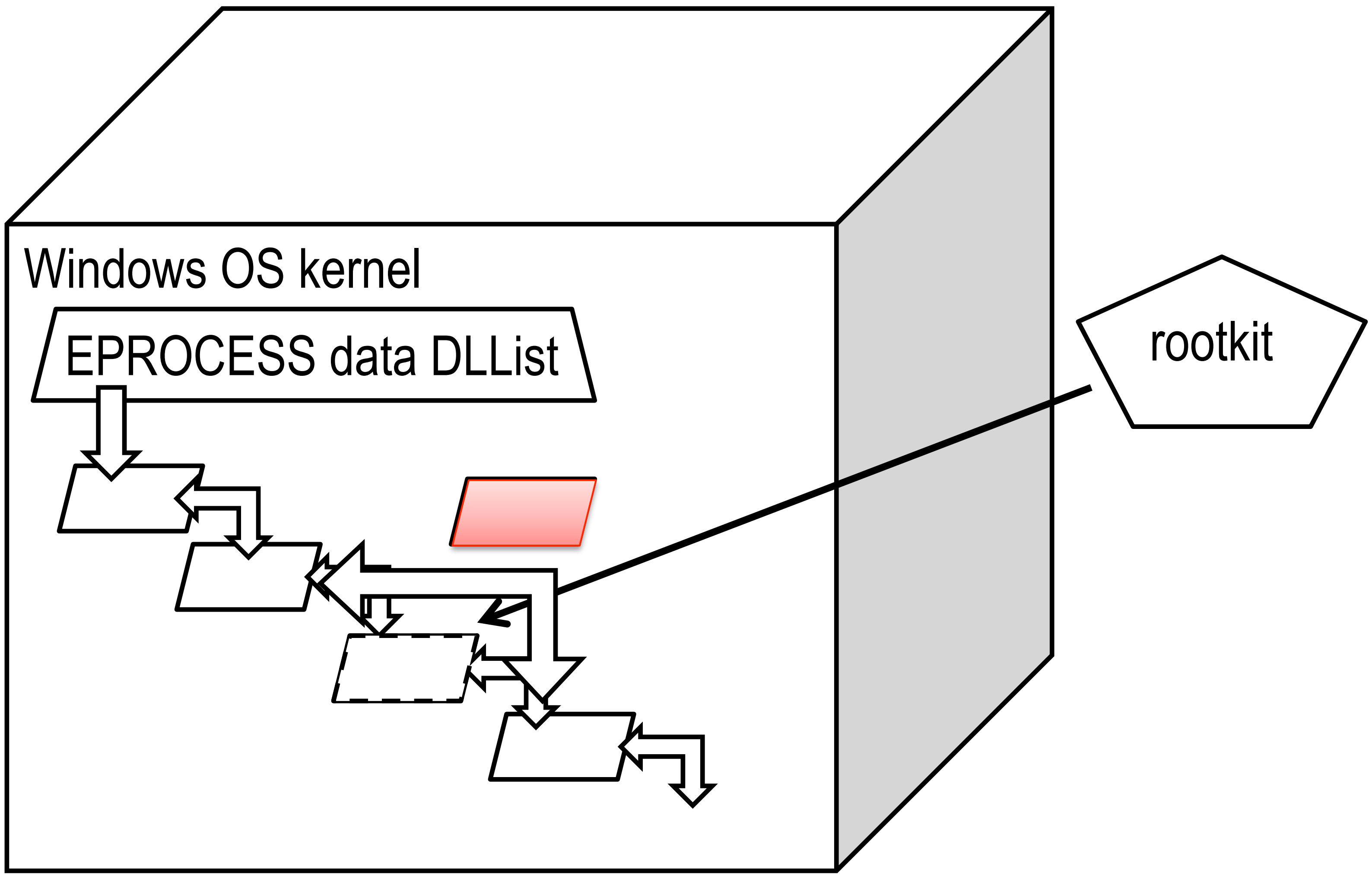
- Part 2
  - CloudSec security appliance for the IaaS cloud platform
  - Points-to analysis of large OS kernel code
  - Kernel object discovery for security engineering

CloudSec

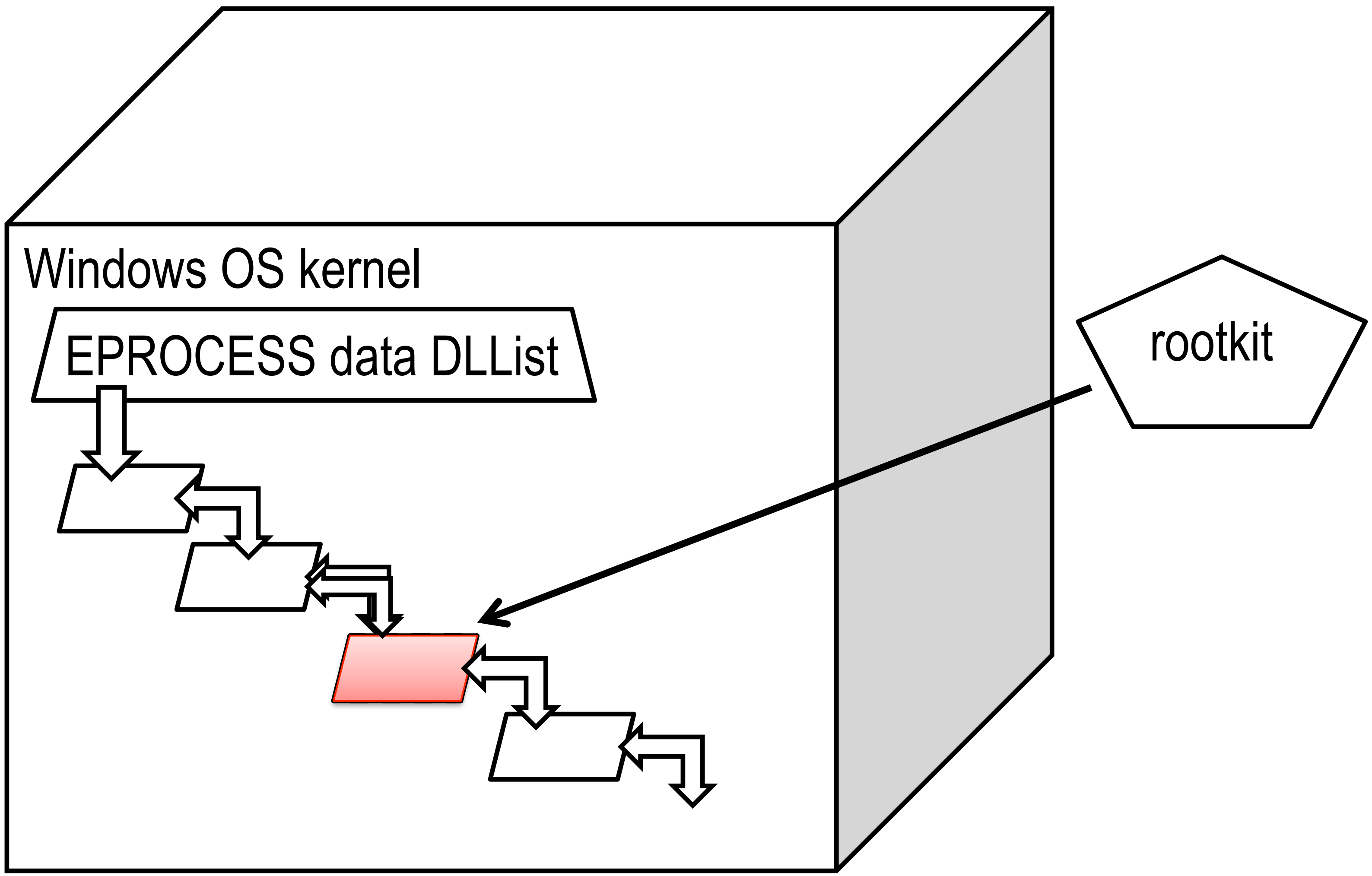
- Problem:
  - ☐ OS kernel rootkits modify data structures to subvert e.g. retarget processing, access data, hide bad processes etc
  - ☐ Most OSes are written in C - heavily use C void pointers, null pointers, casting etc to "mimic" objects
  - ☐ OSs are huge – millions lines of C code
  - ☐ No data structure integrity checking is done by kernel (as its an overhead and not expecting such attacks)
  - ☐ Running security software in virtualised OS e.g. for Cloud computing is problematic (can be compromised)
  - ☐ Virtual Machines (VMs) run on top of a hypervisor layer; compromising hypervisor via root-kit => VMs compromised

  => Serious security holes that need to be addressed

Example 1

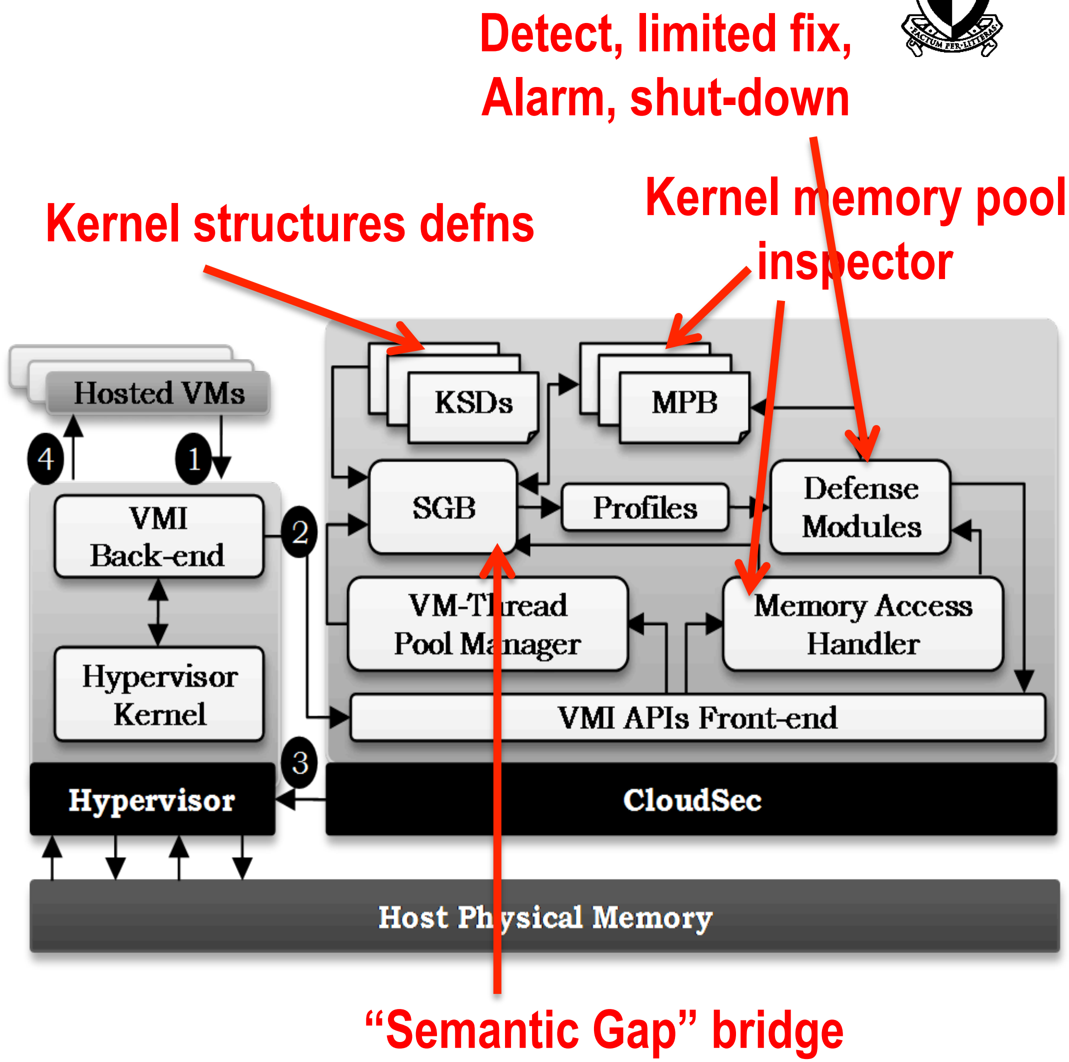


Example 2



CloudSec Architecture

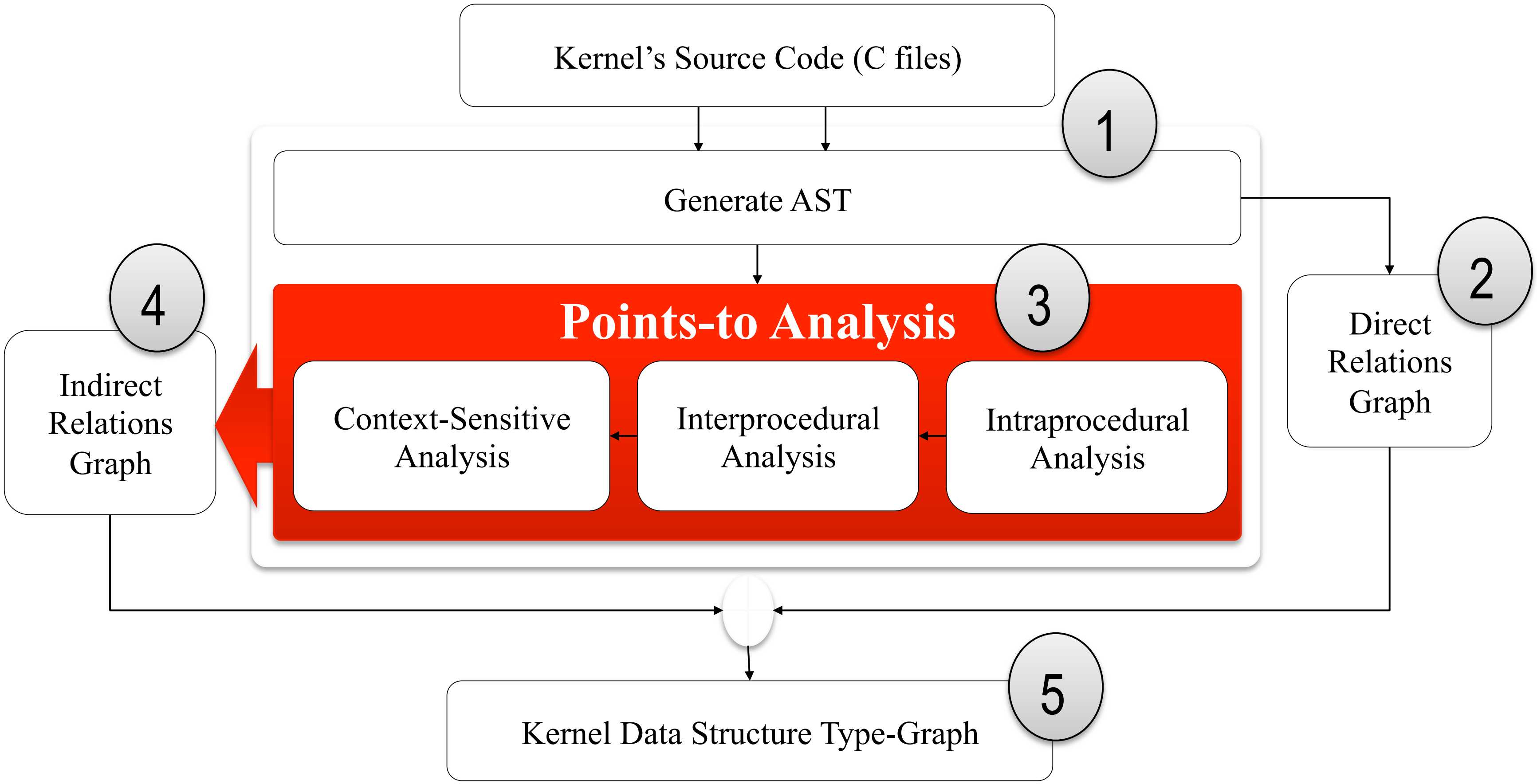
- Back-end
  - ✓ VMWare VMI (Virtual Machine Introspection) APIs
  - ✓ Inspect/control VM's hardware
  - ✓ Enables us to gain control over the hosted VMs to suspend access to VM's hardware, read memory bytes
- Front-end
  - ✓ A set of APIs that allow communication with the back-end
  - ✓ Allows installing triggers (access or timer) on the physical memory pages that need to be monitored



Supporting Technique #1 - KDD

- Need: precise definition of OS kernel data structures
  - BUT: as C-based OSs, one doesn't exist (casts, null pointer refs etc)
- KDD = a new static analysis tool to generate an accurate type graph for any C program
  - Is able to generate a *sound* data definition for large C-based OS *without* any prior knowledge of kernel data layout
  - Disambiguates pointer relations including generic pointers to infer their candidate types & values by performing static points-to analysis on source code
  - New points-to analysis algorithm with interprocedural, context-sensitive and field-sensitive points-to analysis
  - Scales to extremely large C programs that contain millions of lines of code
  - Performs its analysis "off-line" – thus generated type graph can be used by security solutions in on-line security mode (~50 hours for LINUX kernel typing)

KDD Process

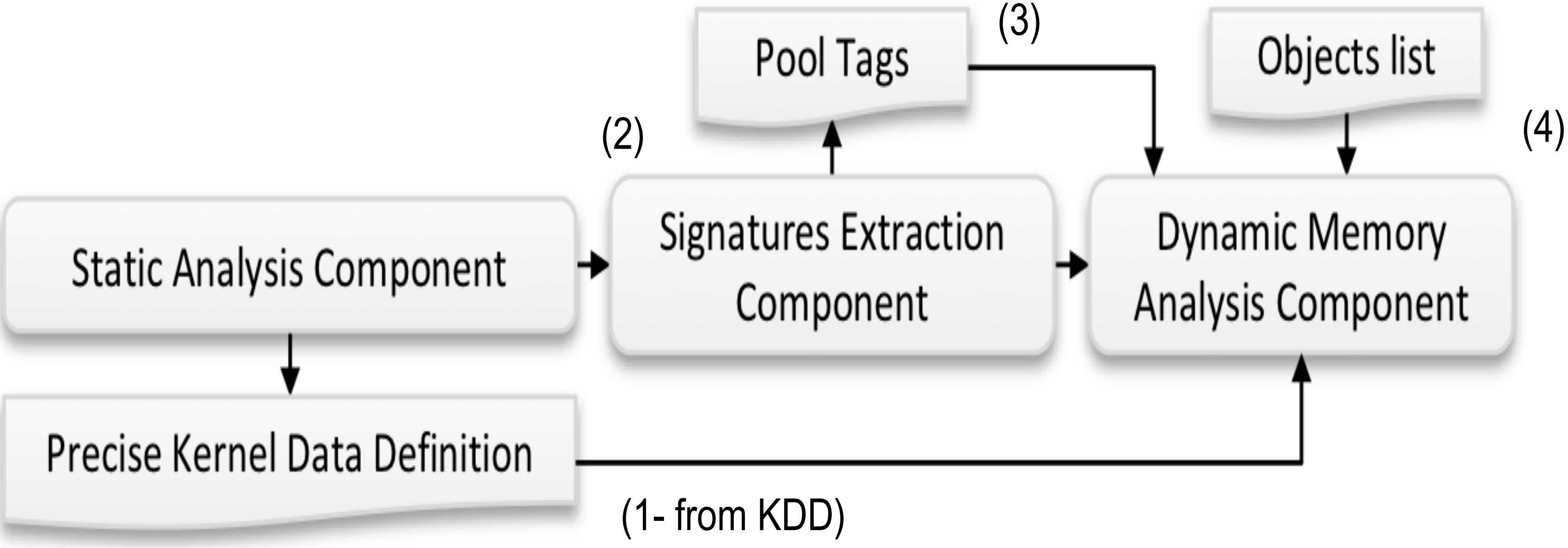


Kernel's Source Code (C files)
1
Generate AST
2
Direct Relations Graph
3
Points-to Analysis
Context-Sensitive Analysis
Interprocedural Analysis
Intraprocedural Analysis
4
Indirect Relations Graph
5
Kernel Data Structure Type-Graph

Supporting Technique #2 - DIGGER

- Problem: in order to protect kernel data structures, need to locate kernel data structures in VM memory – “objects”
  - BUT: this is a challenge – C-based OSs, running in Virtual Machine (must map objects from physical memory bytes)
- DIGGER = a new kernel OS object discovery approach
  - Use VMI to extract memory byes
  - Use special Windows object signatures to locate “objects”
  - Use KDD type graph to “type” the bytes
  - Use discovered objects to identify data structure compromises
- Limited mitigations– raise alarm / “fix” structures / shut down process and/or VM

DIGGER Process



Evaluation - KDD

- Soundness and Precision
  - The points-to analysis algorithm is sound if the points-to set for each variable contains all its actual runtime targets, and is imprecise if the inferred set is larger than necessary
    - Used SPEC2000 and SPEC2006 benchmark suites and other open source C programs
- OS Kernel Analysis
  - WRK (~ 3.5 million LOC) and Linux kernel v3.0.22 (~ 6 million LOC)
    - 28 hours to analyse the WRK and around 47 hours to analysis the Linux kernel.

Benchmark`	LOC	Pointer Inst	Proc	Struct	AST T (sec)	AST M (MB)	AST C (%)	TG T (sec)	TG M (MB)	TG C (%)	P (%)	S (%)
art	1272	286	43	19	22.7	21.5	19.9	73.3	12.3	17.6	100	100
equake	1515	485	40	15	27.5	25.4	20.4	87.5	14.1	21.1	98.6	100
mcf	2414	453	42	22	43.2	41	28.5	14	23	27	97.2	100
gzip	8618	991	90	340	154.2	144.6	70.5	503.3	81.4	68.3	95.1	100
parser	11394	3872	356	145	305.2	191.2	76.7	661.4	107.8	74.3	94.5	100
vpr	17731	4592	228	398	316.1	298.7	80.2	1031.5	163.2	79	NA	100
gcc	222185	98384	1829	2806	3960.5	3756.5	93.5	12962	2200	94	NA	100
sendmail	113264	9424	1005	901	2017.2	1915.1	91.6	6609	1075.0	91.5	NA	100
bzip2	4650	759	90	14	82.3	78.1	45.5	271.6	44.2	42.9	95.9	100

Evaluation – DIGGER vs WinDebug

Table 1. Experimental results of DIGGER and WD on Windows XP 32 bit and 64bit. Memory, paged and nonpaged columns reprsent the size in pages (0x1000 graunrality) of the kernel address space, paged pool and nonpaged pool, repectively. WD and DIG refer to WD's and DIGGER results. FN, FP and FP* denote the false negative, reported false positive and the actual false poitive rates, repectively.

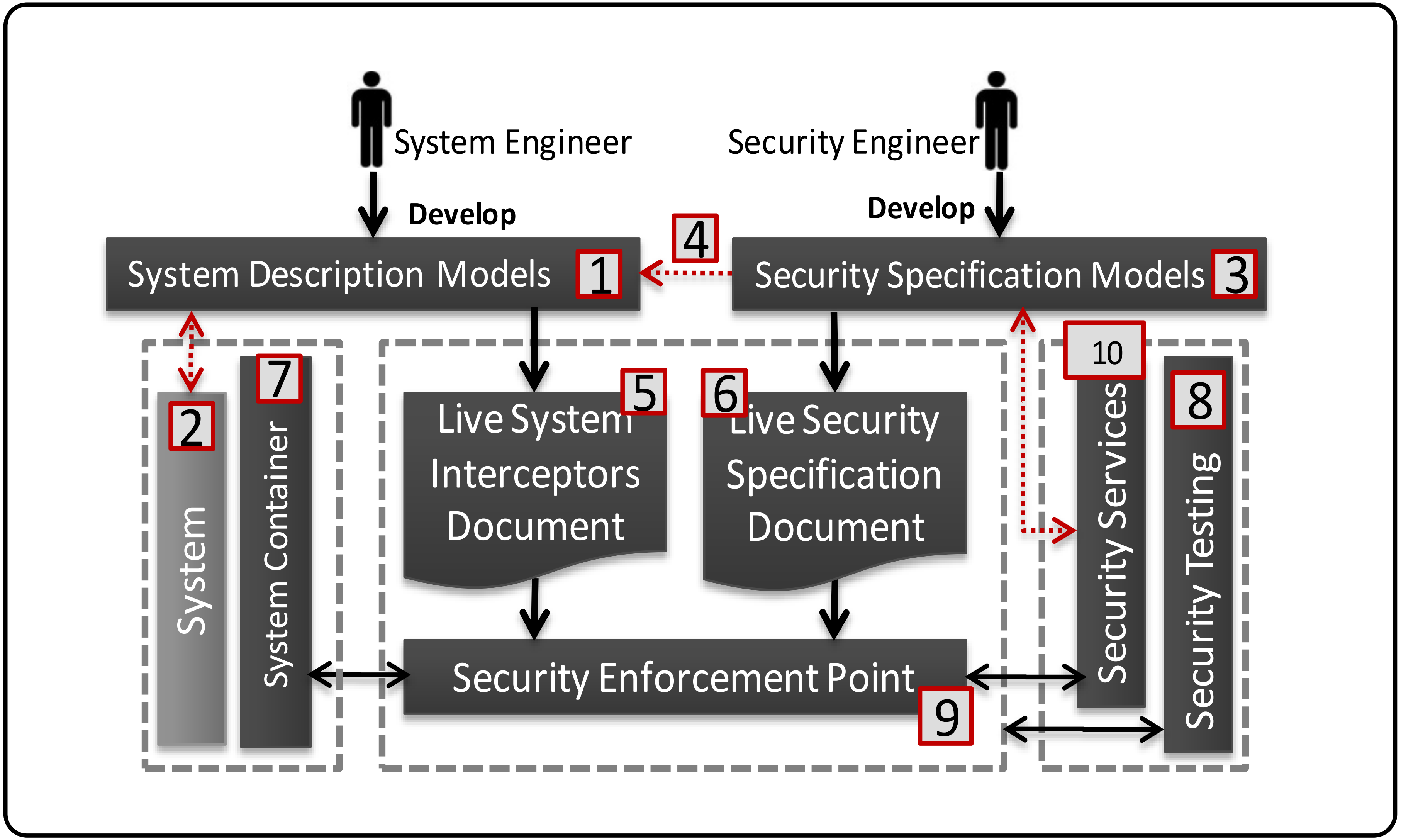
Object	Windows XP 32bit					Windows XP 64bit				
	Memory		Paged		Nonpaged	Memory		Paged		Nonpaged
	915255		27493		11741	1830000		35093		17231
	WD	DIG.	FN %	FP %	FP* %	WD	DIG.	FN %	FP %	FP* %
Process	119	121	0.00	1.65	0.00	125	125	0.00	0.00	0.00
Thread	2032	2041	0.00	0.44	0.00	2120	2121	0.00	0.04	0.00
Driver	243	243	0.00	0.0	0.00	211	211	0.00	0.00	0.00
Mutant	1582	1582	0.00	0.0	0.00	1609	1609	0.00	0.00	0.00
Port	500	501	0.00	0.19	0.00	542	542	0.00	0.00	0.00

- Part 3
  - Model-driven Security Engineering @ Runtime (MDSE@R)
  - Vulnerability Analysis of PaaS, SaaS components
  - Vulnerability mitigation

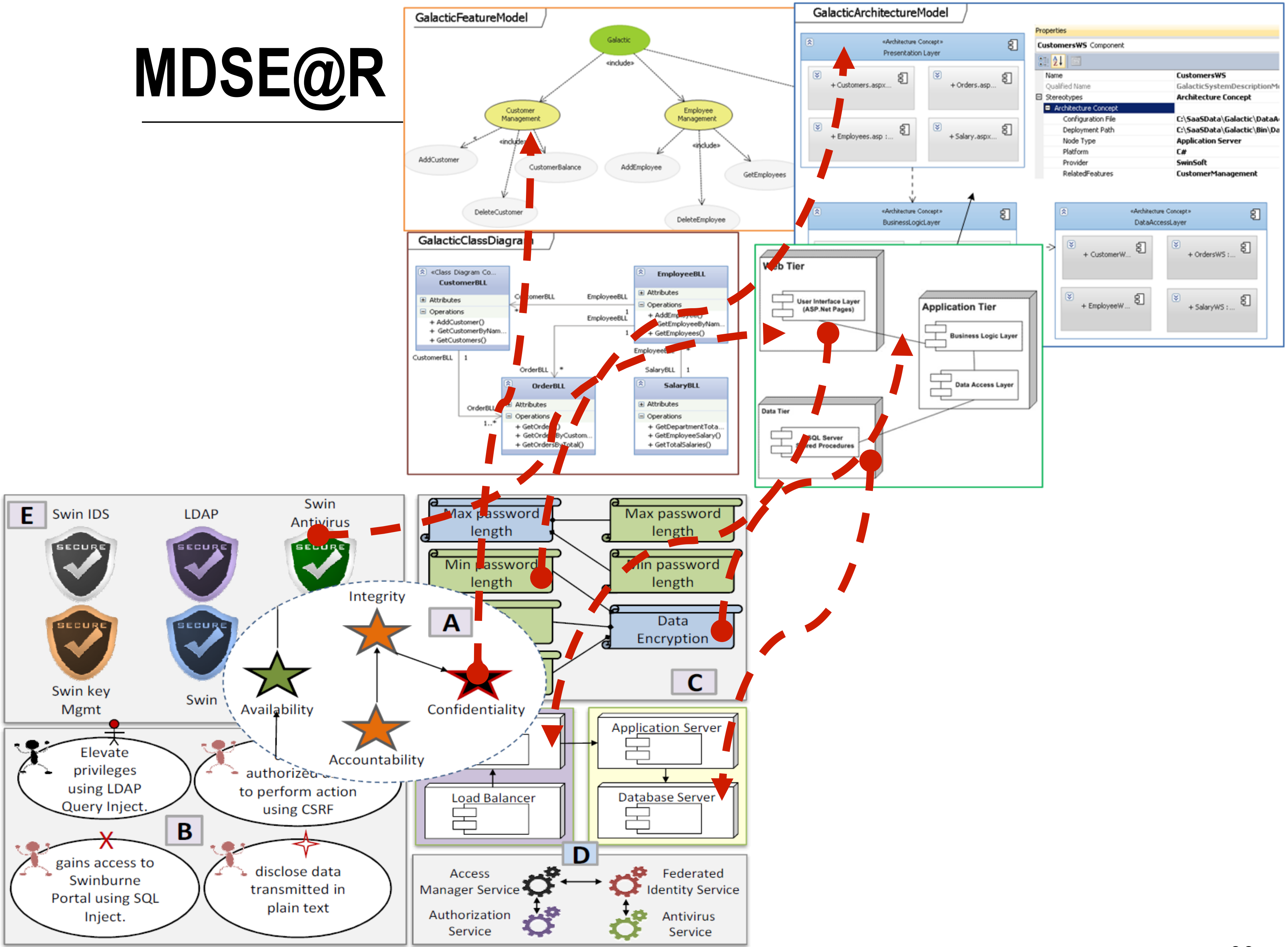
MDSE@R overview

- Problems –
  - How best model security requirements & link to architectural parts of cloud applications?
  - Security requirements set @ design / implementation time – but what if evolve during cloud application deployment?
  - Multi-tenant cloud applications complicate further – what if tennants have different security needs?
- Solution –
  - Model architecture & security; link parts
  - Run-time architecture to update security enforcement of deployed cloud applications

MDSE@R Architecture

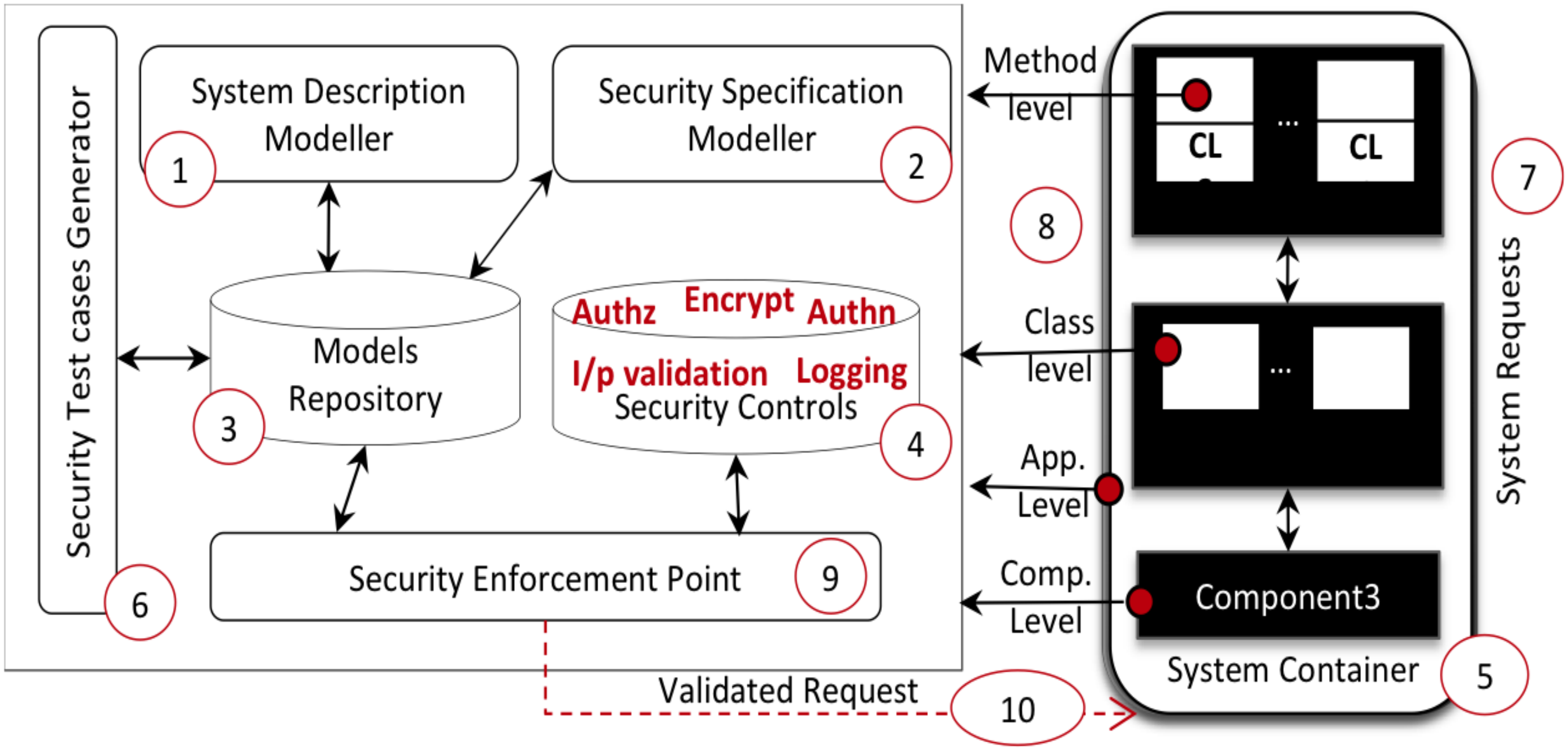


MDSE@R

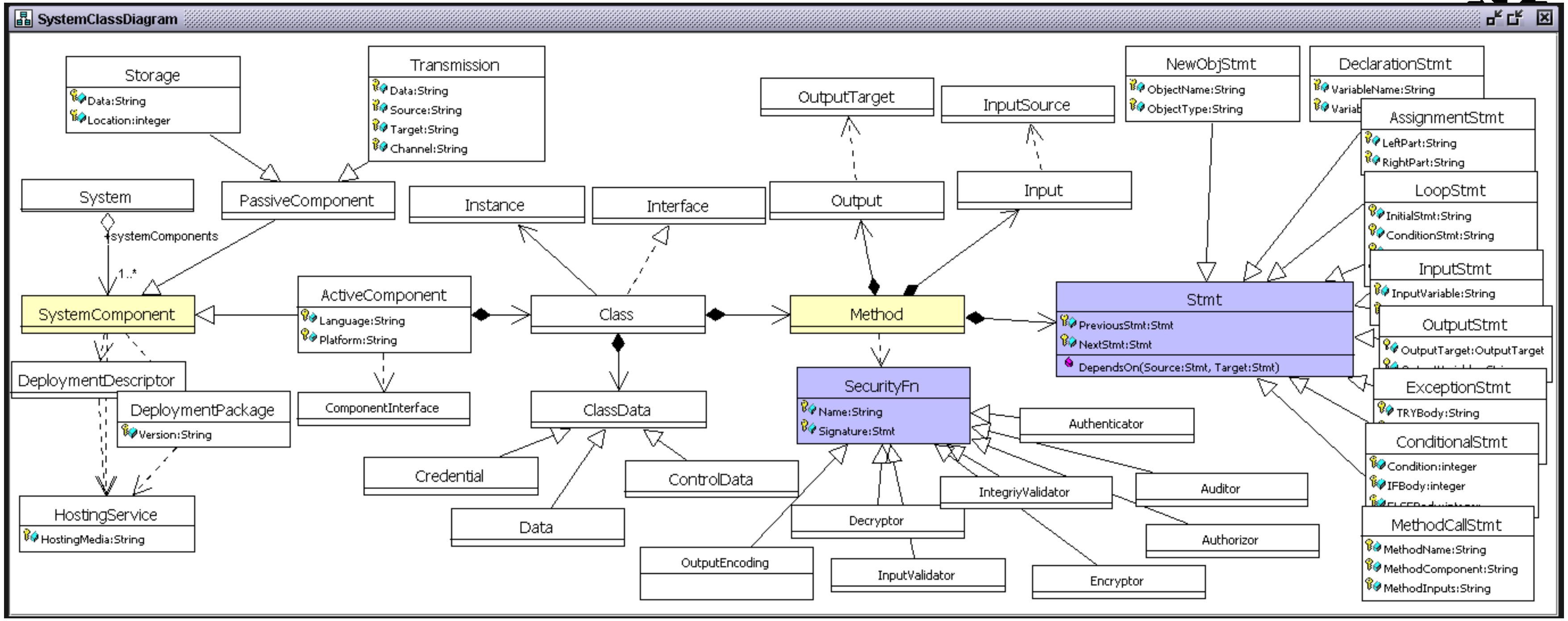


GalacticFeatureModel
Galactic
Customer Management
Employee Management
AddCustomer
CustomerBalance
DeleteCustomer
AddEmployee
GetEmployees
DeleteEmployee
GalacticArchitectureModel
Presentation Layer
BusinessLogicLayer
DataAccessLayer
Properties
CustomersWS Component
Name
CustomersWS
Stereotypes
Architecture Concept
Configuration File
Deployment Path
Node Type
Application Server
Platform
C#
Provider
SwinSoft
RelatedFeatures
CustomerManagement
GalacticClassDiagram
CustomerBLL
EmployeeBLL
OrderBLL
SalaryBLL
Web Tier
User Interface Layer (ASP.Net Pages)
Application Tier
Business Logic Layer
Data Access Layer
Data Tier
SQL Server Stored Procedures
E
Swin IDS
LDAP
Swin Antivirus
Swin key Mgmt
Swin
Integrity
Availability
Confidentiality
Accountability
A
Max password length
Min password length
Data Encryption
C
Elevate privileges using LDAP Query Inject.
authorized to perform action using CSRF
B
gains access to Swinburne Portal using SQL Inject.
disclose data transmitted in plain text
Load Balancer
Application Server
Database Server
D
Access Manager Service
Federated Identity Service
Authorization Service
Antivirus Service

MDSE@R Run-time securing

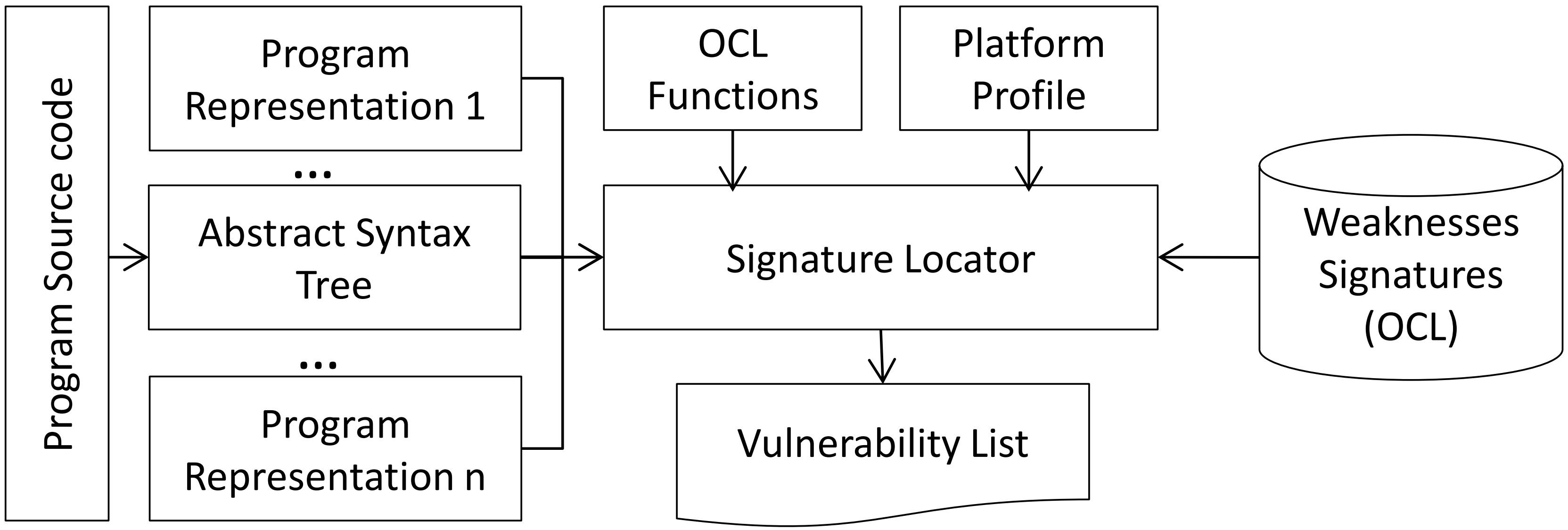


Support technique #1:Vulnerability Analysis



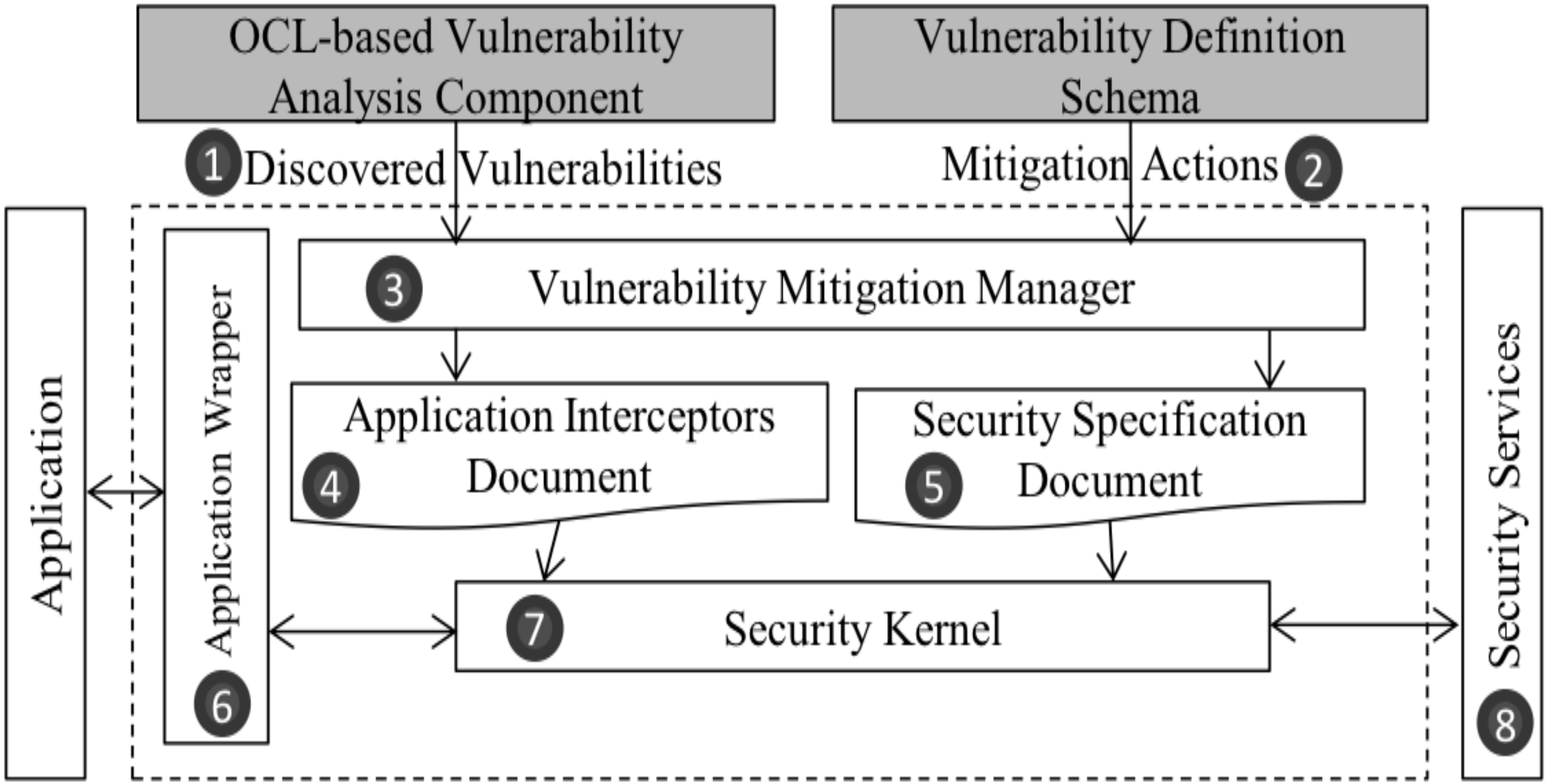
Vul.	Vulnerability Signature
SQLI	Method.Contains(S : MethodCall | S.FnName = “ExecuteQuery” AND S.Arguments.Contains(X : IdentifierExpression | X.Contains(InputSource)))
XSS	Method.Contains(S : AssignmentStatement | S.RightPart.Contains(InputSource) AND S.LeftPart.Contains(OutputTarget))
Improper Authn.	Method.IsPublic == true AND Method.Contains(S : MethodCall | S.IsAuthenitcationFn == true AND S.Parent == IFElseStmt AND S.Parent.Condition.Contains(InputSource))
Improper Authz.	Method.IsPublic == true AND Method.Contains(S : Expression | S.Contains(X: InputSource | X.IsSanitized == False OR X.IsAuthorized == False)

Analyser



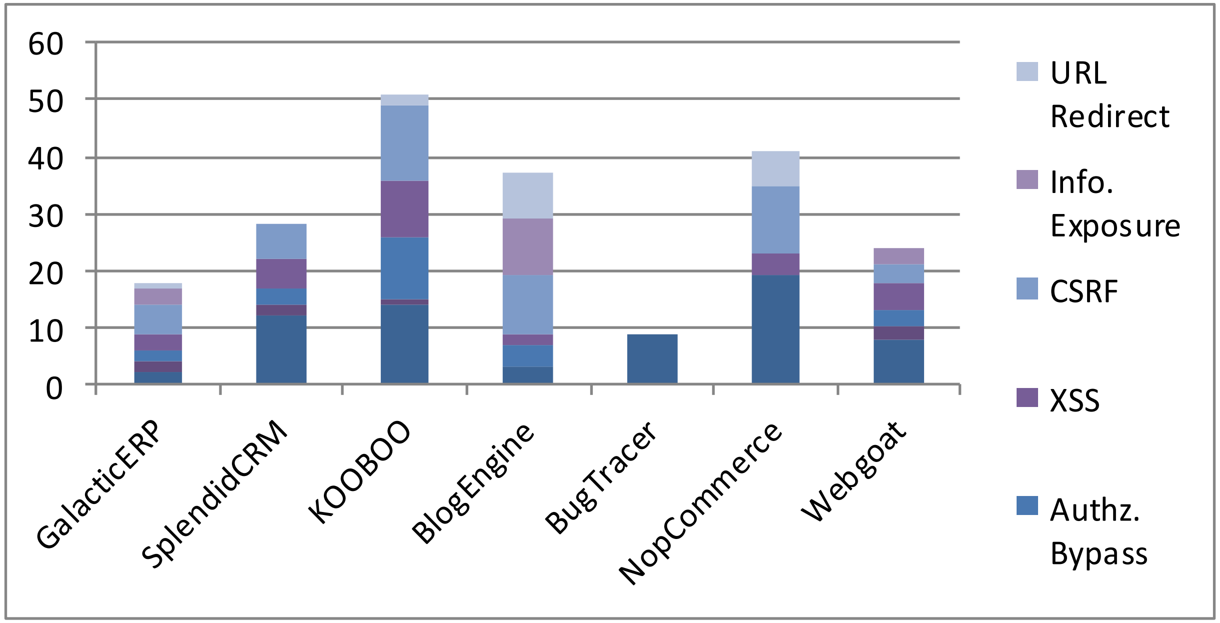
Program Source code
Program Representation 1
...
Abstract Syntax Tree
...
Program Representation n
OCL Functions
Platform Profile
Signature Locator
Weaknesses Signatures (OCL)
Vulnerability List

Support tech. #2: Vulnerability Mitigation



Evaluation – Vulnerability Analysis

Benchmark	Downloads	KLOC	Files	Comps	Classes	Method
BlogEngine	>46,000	25.7	151	2	258	616
BugTracer	>500	10	19	2	298	223
Galactic	-	16.2	99	6	101	473
KOOBOO	>2,000	112	1178	13	7851	5083
NopCommerce	>10 Rel.	442	3781	8	5127	9110
SplendidCRM	>400	245	816	7	6177	6107



Evaluation – MDSE@R

Table 1: Results of validating MDSE@R against Group-1 and Group-2 applications

Benchmark Applications		Statistics			Security Attributes				
		KLOC	Files	Classes	Authn	Authz.	I/P Valid.	Audit	Crypto.
G1	Galactic	16.2	99	101	F, C, S, M				
	PetShop	7.8	15	25	F, C, S, M				
G2	Splendid	245	816	6177	C, S, M				(C, S, M)*
	KOOBOO	112	1178	7851	C, S, M				(C, S, M)*
	NopComm	442	3781	5127	C, S, M				(C, S, M)*
	BlogEngine	25.7	151	258	C, S, M				(C, S, M)*
	BugTracer	10	19	298	C, S, M				(C, S, M)*
	TinyERP	6	20	22	C, S, M				(C, S, M)*

F: Security attribute successfully applied on feature level & propagated to lower entities

C:Security attribute successfully applied on component level & propagated to lower entities

S:Security attribute succesfully applied on classes **M**:Security attribute can be applied on method level

- Part 4
  - Future directions
  - Summary

Current & Future Work

- Monitoring of security of cloud apps @ run-time – what metrics? How? What do if problems detected??
- Applying vulnerability analysis to detect e.g. performance anti-patterns, energy anti-patterns
- CloudSec++ mitigations when attacks detected
- Points-to analysis enhancements – accuracy, cloud-deployment ☺

Summary

- Interested in addressing several challenging problems with cloud application and platform security
- CloudSec – security appliance for virtualised platforms
- MDSE@R – model-driven approach to security modelling and enforcement
- Various supporting techniques interesting research in their own right: vulnerability analysis via OCL signatures; customer-managed security preferences; points-to analysis; OS kernel object discovery for virtualized servers; re-aspects updating of existing systems in sophisticated ways

References

- Almorsy, M., Ibrahim, A., Grundy, J.C., Adaptive Security Management in SaaS Applications, Chapter 8 in Security, Privacy and Trust in Cloud Systems, Springer, 2013.
- Almorsy, M., Grundy, J.C. and Ibrahim, A., Automated Software Architecture Security Risk Analysis Using Formalized Signatures, 2013 IEEE/ACM International Conference on Software Engineering (ICSE 2013), San Franciso, May 2013, IEEE CS Press
- Almorsy, M., Grundy, J.C. and Ibrahim, A. Supporting Automated Vulnerability Analysis using Formalized Vulnerability Signatures, 27th IEEE/ACM International Conference on Automated Software Engineering (ASE 2012), Sept 3-7 2012, Essen, Germany, ACM Press.
- Ibrahim, A., Hamlyn-Harris, J., Grundy, J.C. and Almorsy, M., Supporting Operating System Kernel Data Disambiguation using Points-to Analysis, 27th IEEE/ACM International Conference on Automated Software Engineering (ASE 2012), Sept 3-7 2012, Essen, Germany, ACM Press.
- Almorsy, M., Grundy, J.C. and Ibrahim, I., VAM-aaS: Online Cloud Services Security Vulnerability Analysis and Mitigation-as-a-Service, 2012 International Conference on Web Information Systems Engineering (WISE 2012), Nov 28-30 2012, Paphos, Cyprus, LNCS, Springer.
- Ibrahim, A., Hamlyn-Harris, J., Grundy, J.C., Almorsy, M., Identifying OS Kernel Runtime Objects for Run-time Security Analysis, 2012 International Conference on Network and System Security (NSS 2012), Fujian, China, Nov 21-23 2012, LNCS, Springer.