# MULTI-TENANT CLOUD APPLICATION RUN-TIME SECURITY MONITORING AND ANALYSIS

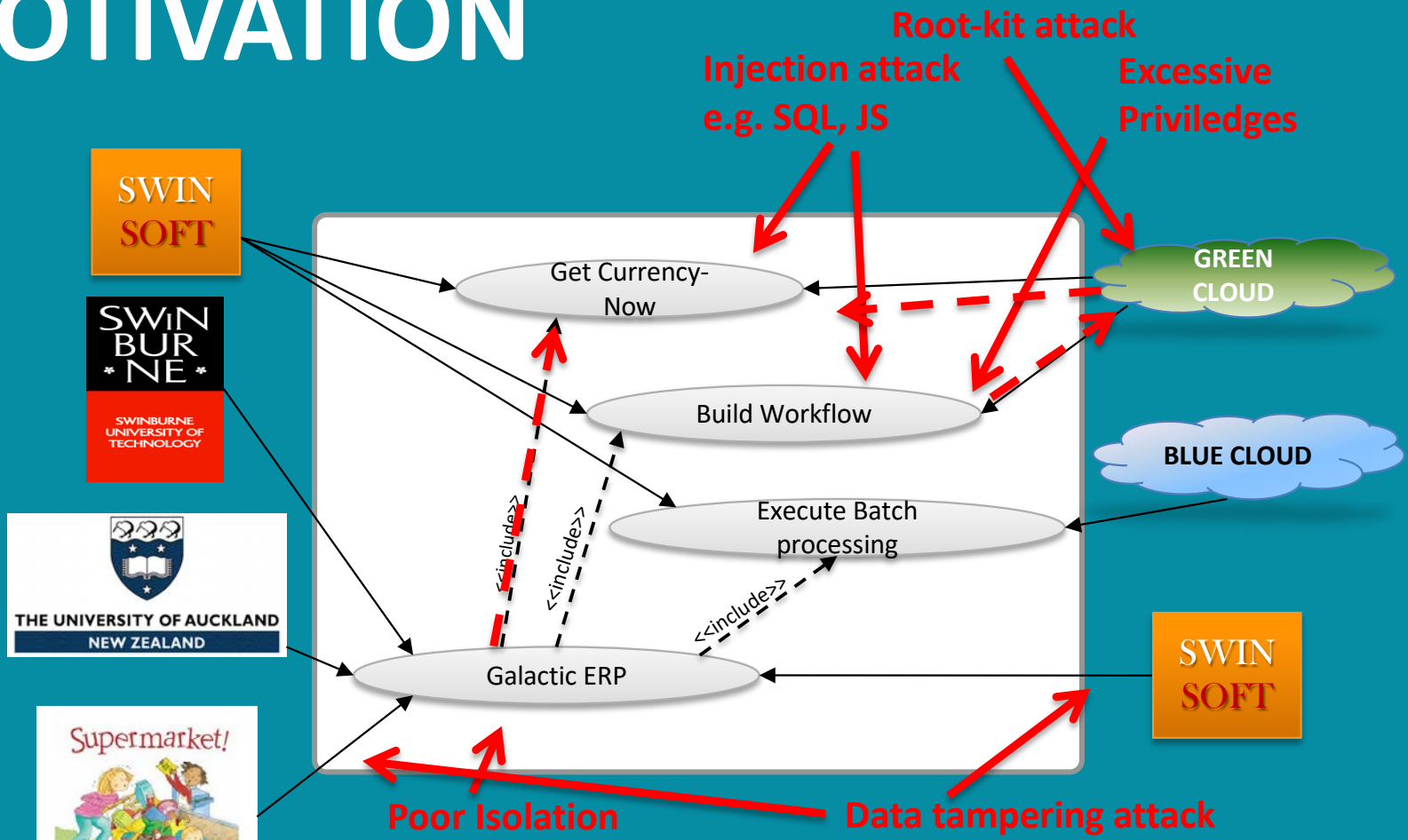
Prof John Grundy
PVC ICT Innovation & Translation
Professor of Software Engineering

DEAKIN UNIVERSITY AUSTRALIA
Worldly

# OUTLINE

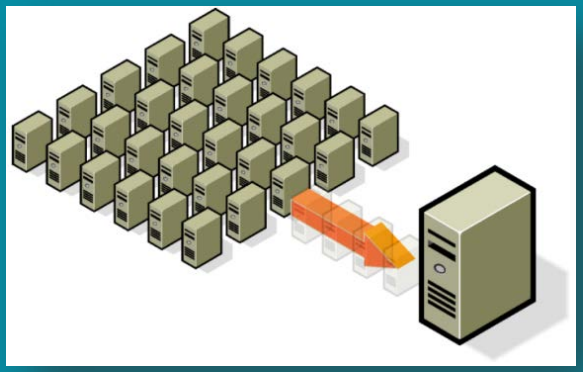
- Motivating example
- CloudSec – security appliance for cloud VMs
- SMART - (static; to-be dynamic) vulnerability analysis
- Log / metric correlation analysis (dynamic analysis)
- Run-time cloud monitoring via generated probes (static & dynamic)
- Mitigation via run-time update (models @ run-time approach)
- Tenant-specified security requirements
- Future directions…

# 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 EC2
- Platform as as Service (PaaS) e.g. Google App Engine
- Software as a Service (SaaS) e.g. SalesForce.com
- Multi-tenant applications sharing IaaS, Paas, SaaS…

DEAKIN UNIVERSITY AUSTRALIA Worldly

# KEY SECURITY PROBLEMS W CLOUD MODEL

- IaaS:
  - Cloud providers don't know what is running on their VMs
  - Cloud users don't know what other apps running / infrastructure security policies
- PaaS:
  - Design-time focus of security solutions BUT security needs emerge @ run-time esp with multi-tenant, extensible SaaS applications
  - Lack of integration of security / cloud application architecture
- SaaS:
  - Different tenant security needs for same SaaS application
  - Evolving tenant needs / limited (no?) tenants involvement in security configuration

# OUR APPROACH(ES) TO ADDRESS…

- IaaS protection:
  - **(1) 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 (I won't say much about this today)
  - Supported by (2) **SMART** - **vulnerability analysis** & (5) **run-time mitigation as-a-service, re-aspects**
  - **(3) Log file / runtime cloud metric correlation analysis**
  - **(4) Monitoring/metric probe generation**

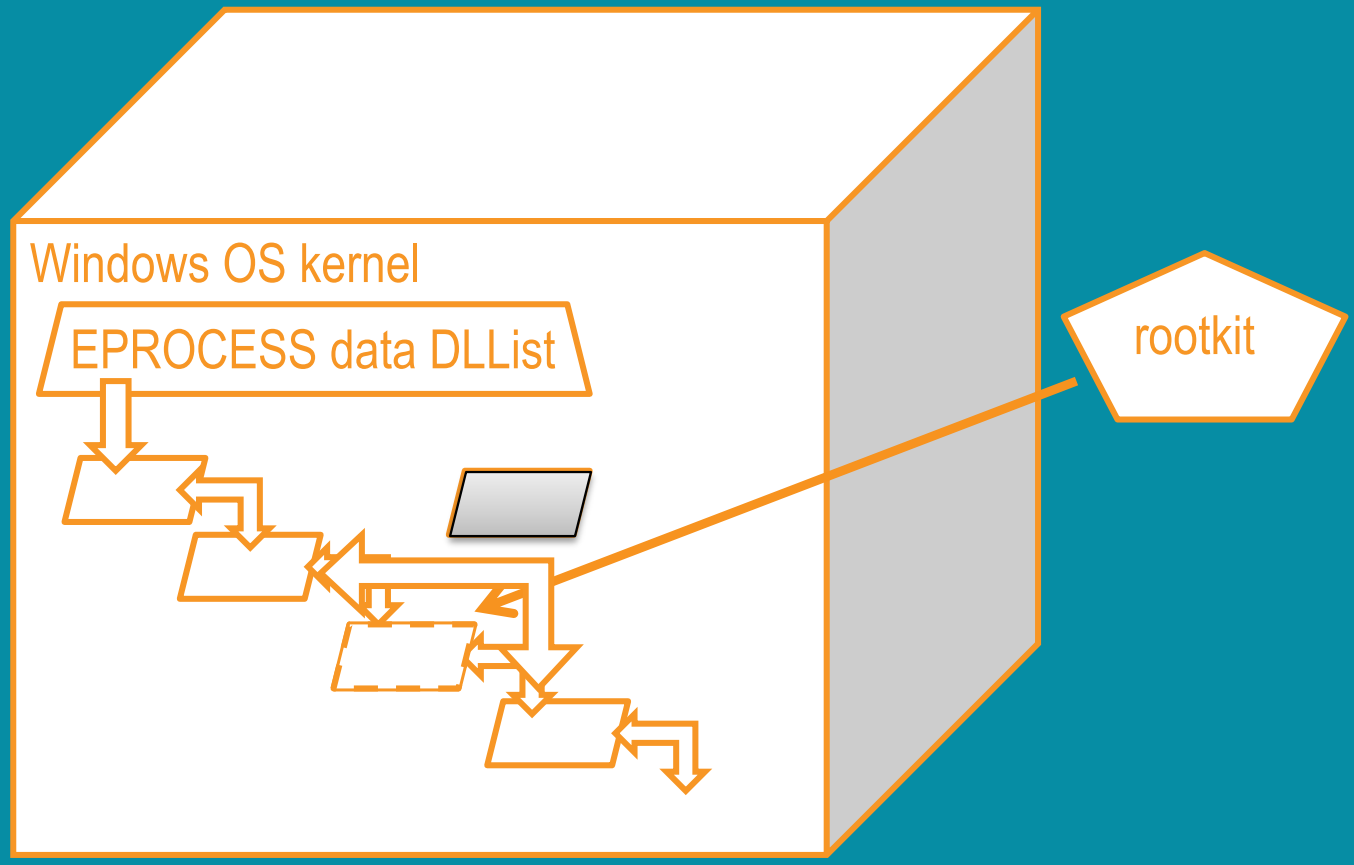
- SaaS:
  - **(6) TOSSMA** – cloud consumer security management console
  - SMURF – multi-tenant re-engineering via re-aspects

# TECHNIQUE #1 - CLOUDSEC

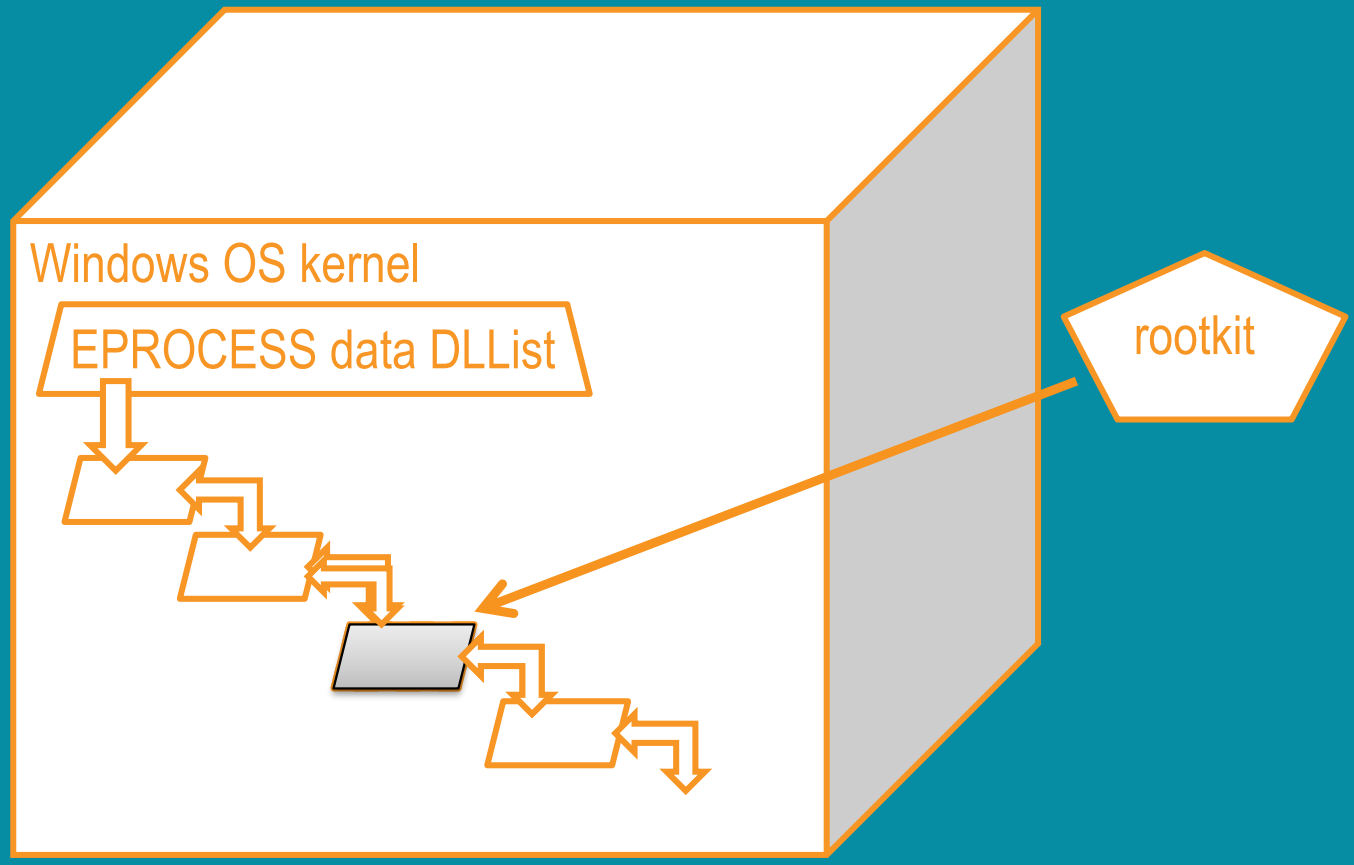
- Problem:
  - OS kernel rootkits modify data structures to subvert e.g. retarget processing, access data, hide bad processes etc
  - Most OSes are written in C - heavily use C void pointers, null pointers, casting etc to “mimic” objects
  - OSs are huge – millions lines of C code
  - No data structure 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

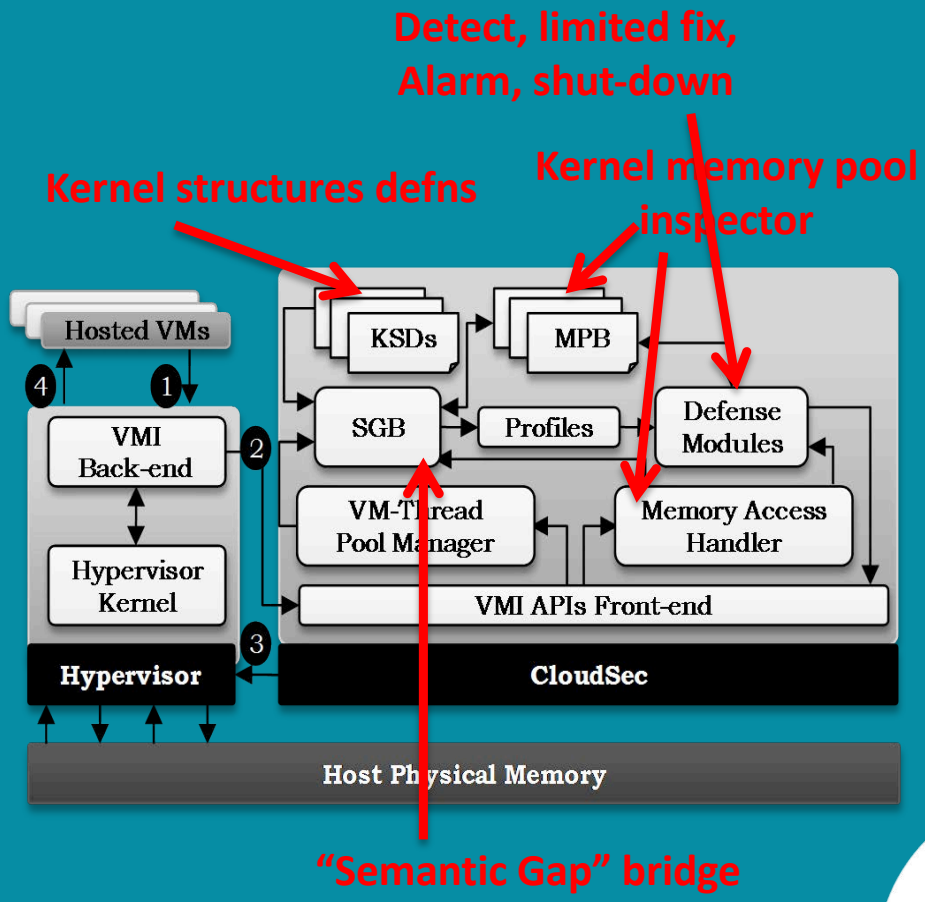


# EXAMPLE 2



# 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 inter-procedural, 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)

# 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

# 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 |

# TECHNIQUE #2 – VULNERABILITY ANALYSIS

- Part of larger “model-driven security engineering @ run-time” (MDSE@R) platform (another talk for another day… ☺)
- Formalise the OWSAP and CAPEC database of security vulnerabilities into “signatures” ; search for these in code/models
- Handles code vulnerability detection and design, architecture vulnerability detection & security “metrics”
- Some vulnerabilities have a “mitigation” – some can apply at run-time using MDSE@R platform (run-time security enforcement) and/or our ”Re-aspects” framework (run-time .NET code updating)

# EXAMPLES…

```
Public bool LogUser(string username, string password) {
    string query = "SELECT username FROM Users WHERE
    UserID ='" username " ' AND Password = '" +  password + "'";
```

**Figure 2. A code snippet vulnerable to SQLI attack**

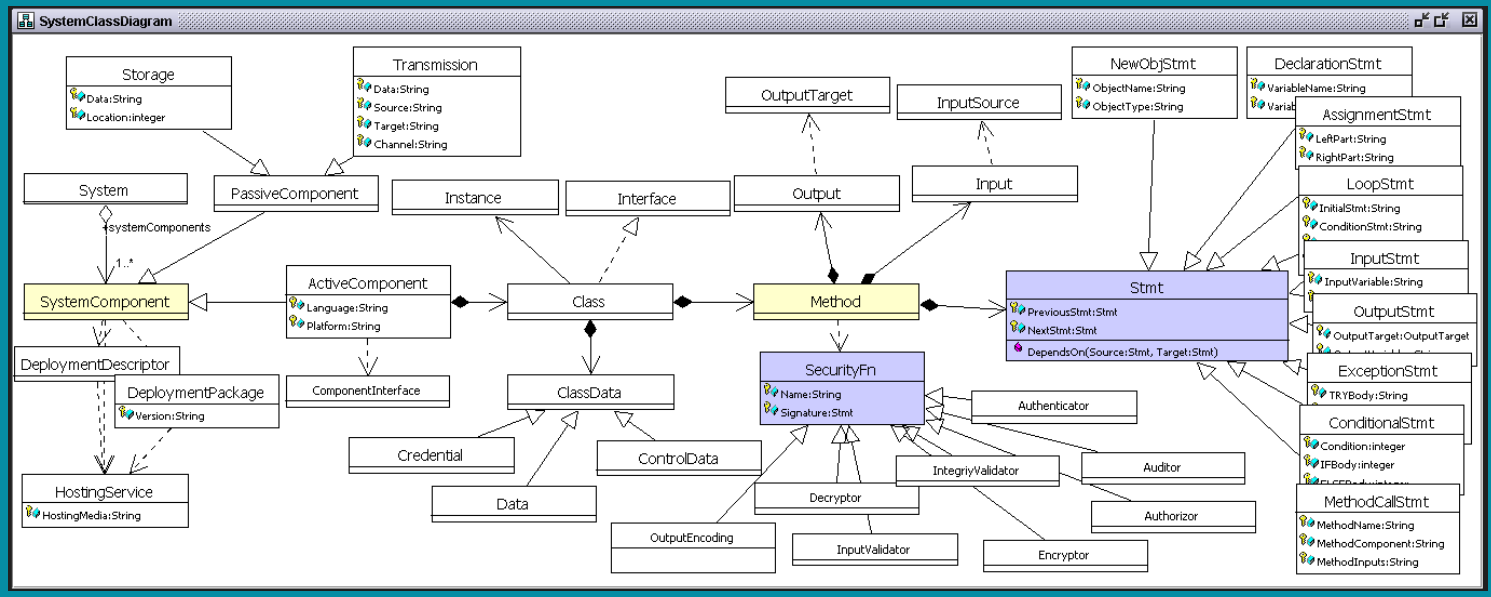
```
if( Request.Cookies["Loggedin"] != true ) {
        if(  !AuthenticateUser(Request.Params["username"],
                                Request.Params["password"] ) )
            throw new Exception("Invalid user");
}
DoAdministrativeTask();
```

**Figure 3. A code snippet vulnerable to authentication Bypass**

```
if( !AuthenticateUser( Request.Params["username"],
                        Request.Params["password"] ) )
        throw new Exception("Invalid user");
updateCustomerBalance(Request.QueryString["custID"], nBalance);
```

**Figure 4. A code snippet vulnerable to improper authz**

# SMART VULNERABILITY ANALYSIS TOOL

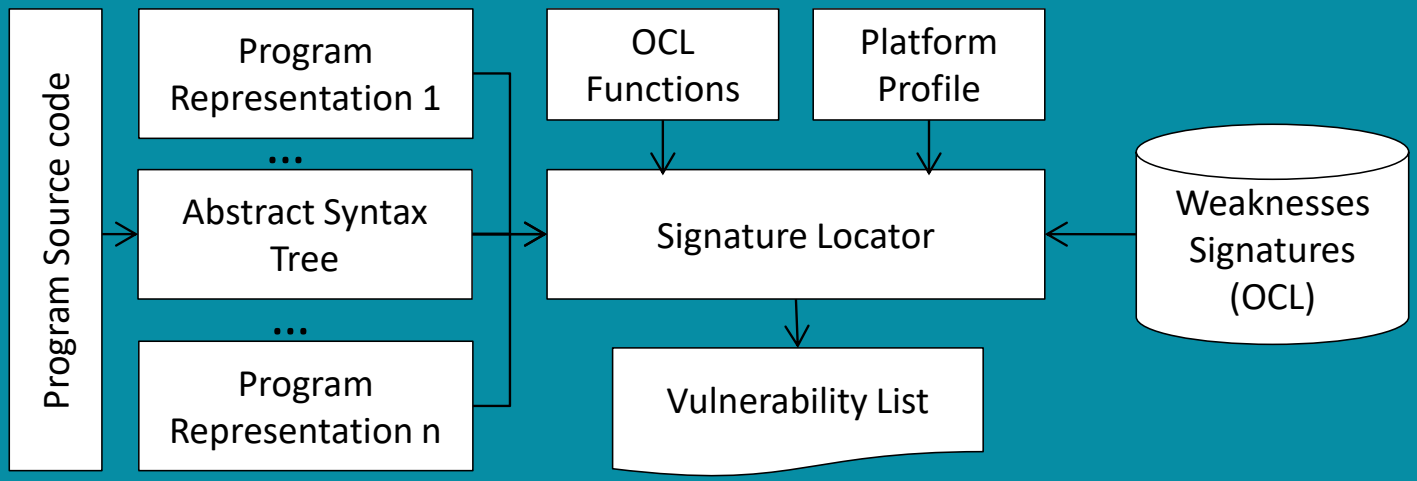



| Vul. | Vulnerability Signature (Simplified!!) |
|---|---|
| SQLI | Method.Contains( S : MethodCall \| S.FnName = "ExecuteQuery" AND S.Arguments.Contains( X : IdentifierExpression \| X.Contains(InputSource))) |
| XSS | Method.Contains(S : AssignmentStatement \| S.RightPart.Contains(InputSource) AND 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) |



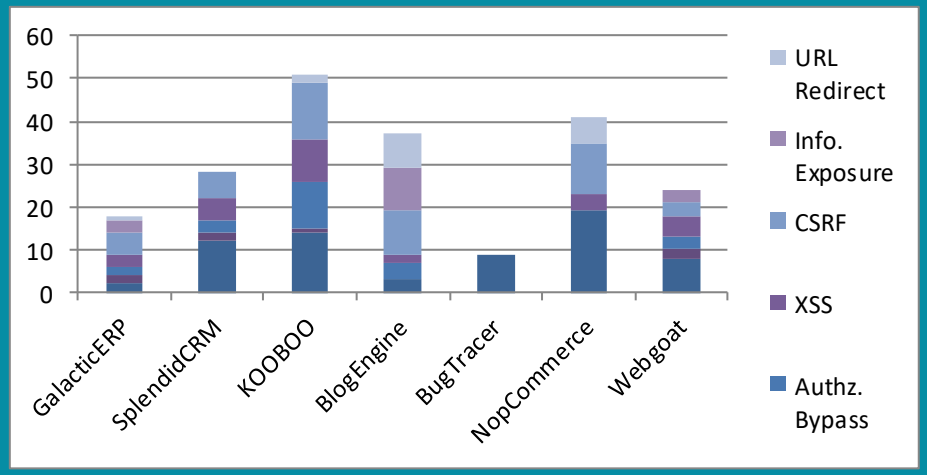
DEAKIN UNIVERSITY AUSTRALIA Worldly

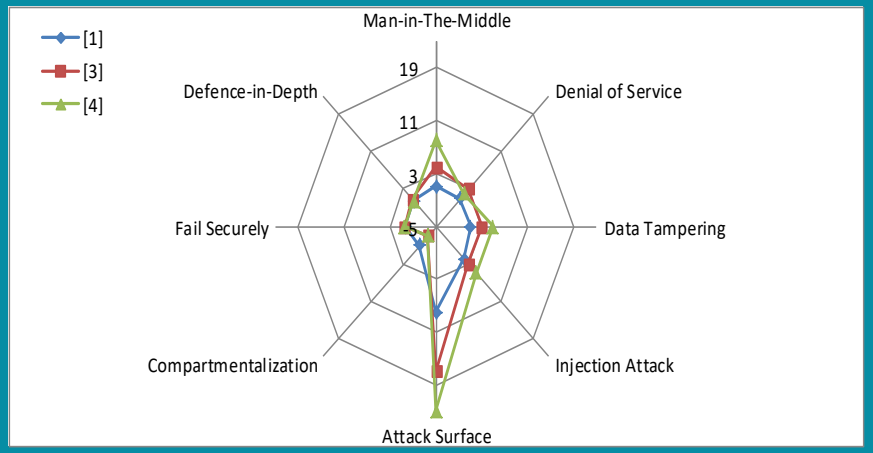
# (STATIC) ANALYSER



# EVALUATION – VULNERABILITY ANALYSIS (STATIC)

| 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 |

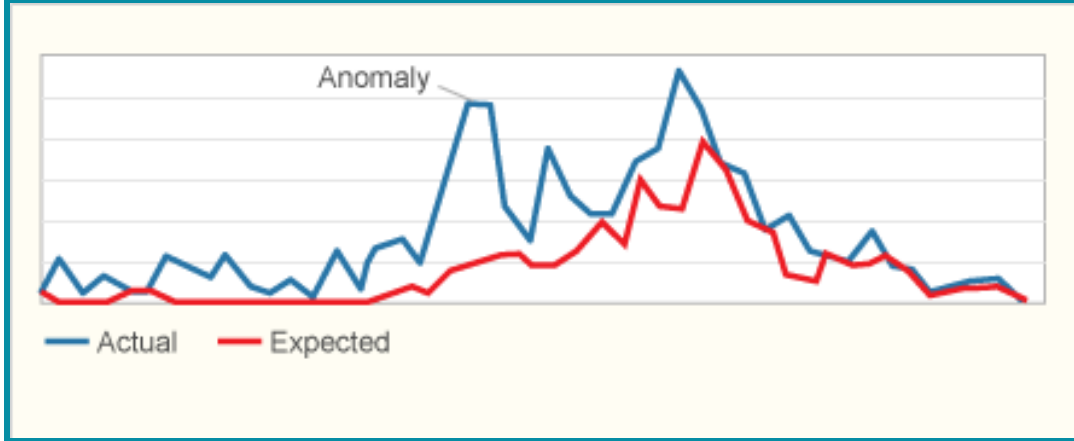


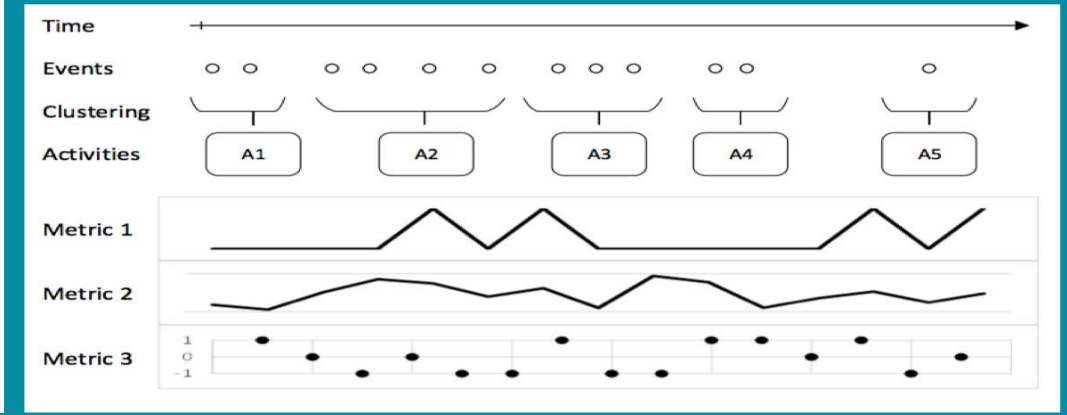



# TECHNIQUE #3 – LOG FILE/CLOUD PAAS METRIC ANALYSIS (DYNAMIC ANALYSIS)

- Applied to large scale cloud operations e.g. rolling upgrade
- These complex operations often fall over due to various issues encountered during the operation
- Detecting – and fixing is (very) hard
- Our approach – take log file & monitor cloud metrics – do correlation analysis to determine occurrence of cloud operation exceptions
- Aim to generate assertions / monitors to determine proactively different cloud operation exceptions
- Lots of challenges – detail in logs; log collection timings; access to detailed cloud metrics; metric capture frequency and accuracy;

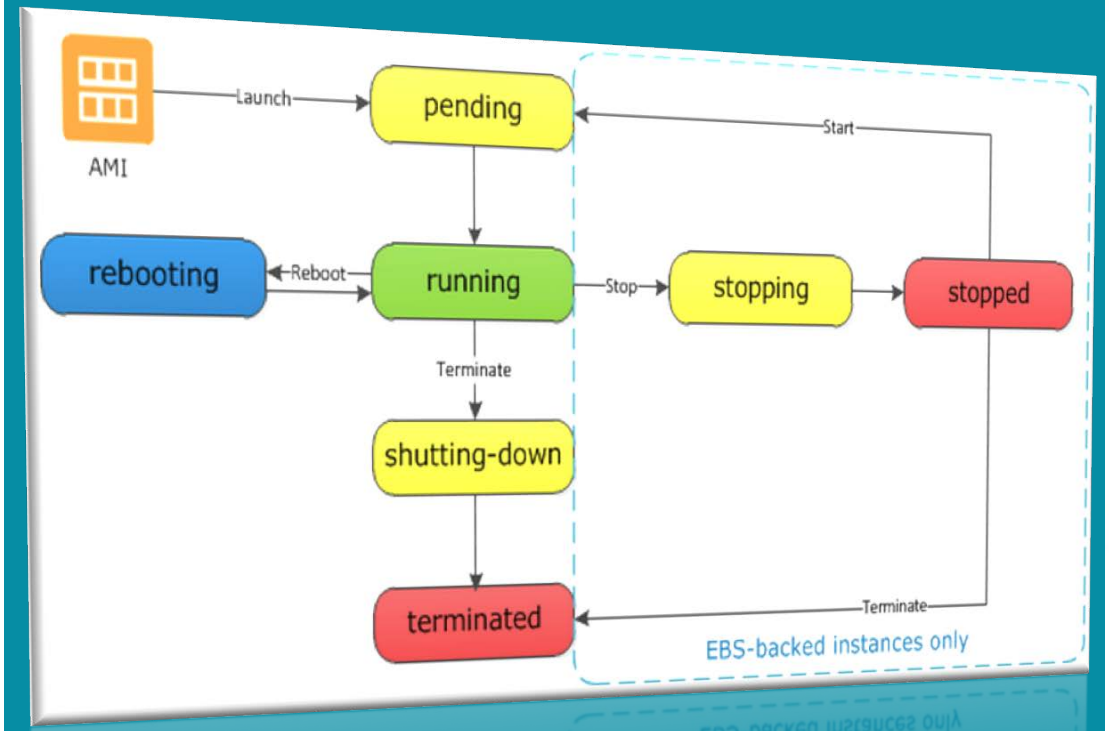
# ANOMALY DETECTION

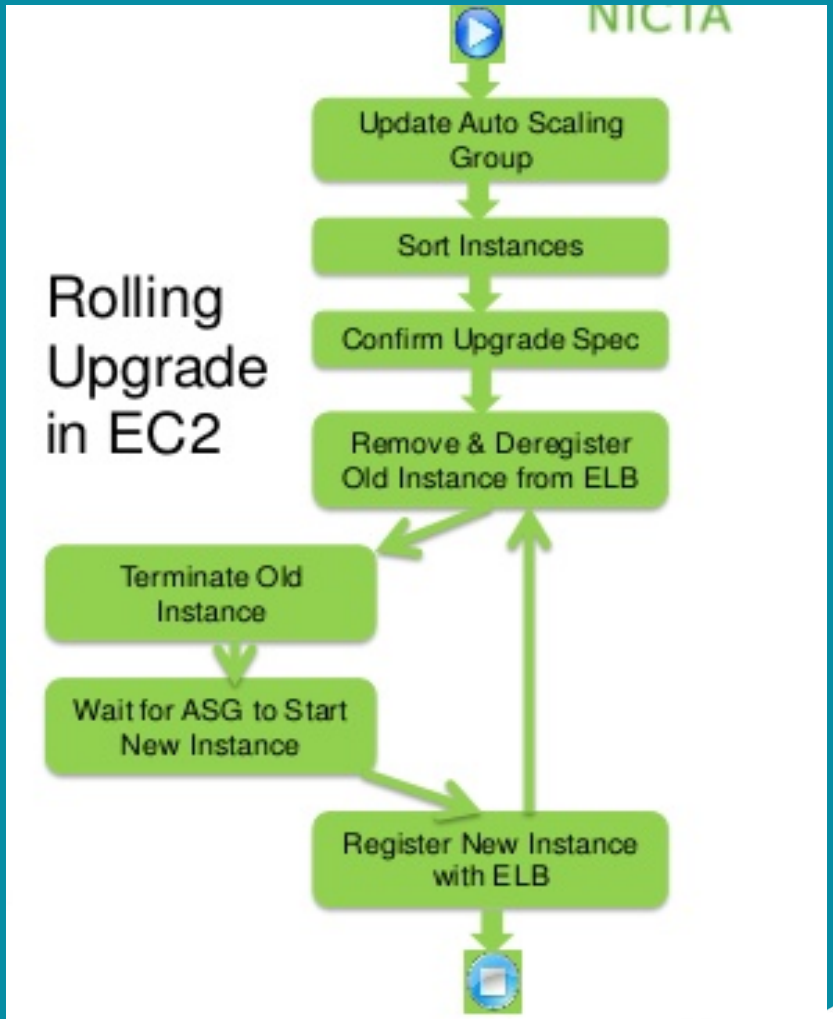


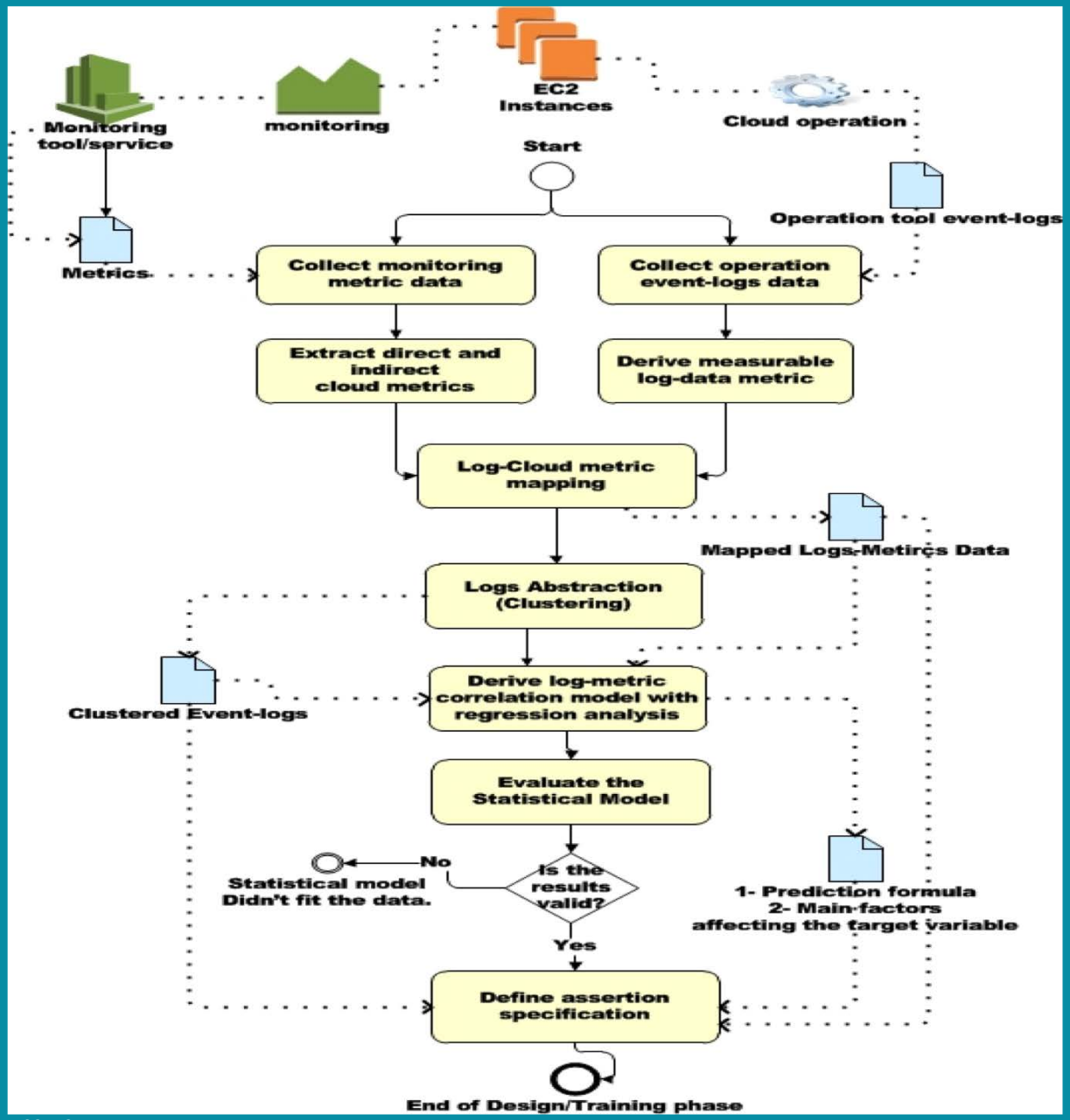



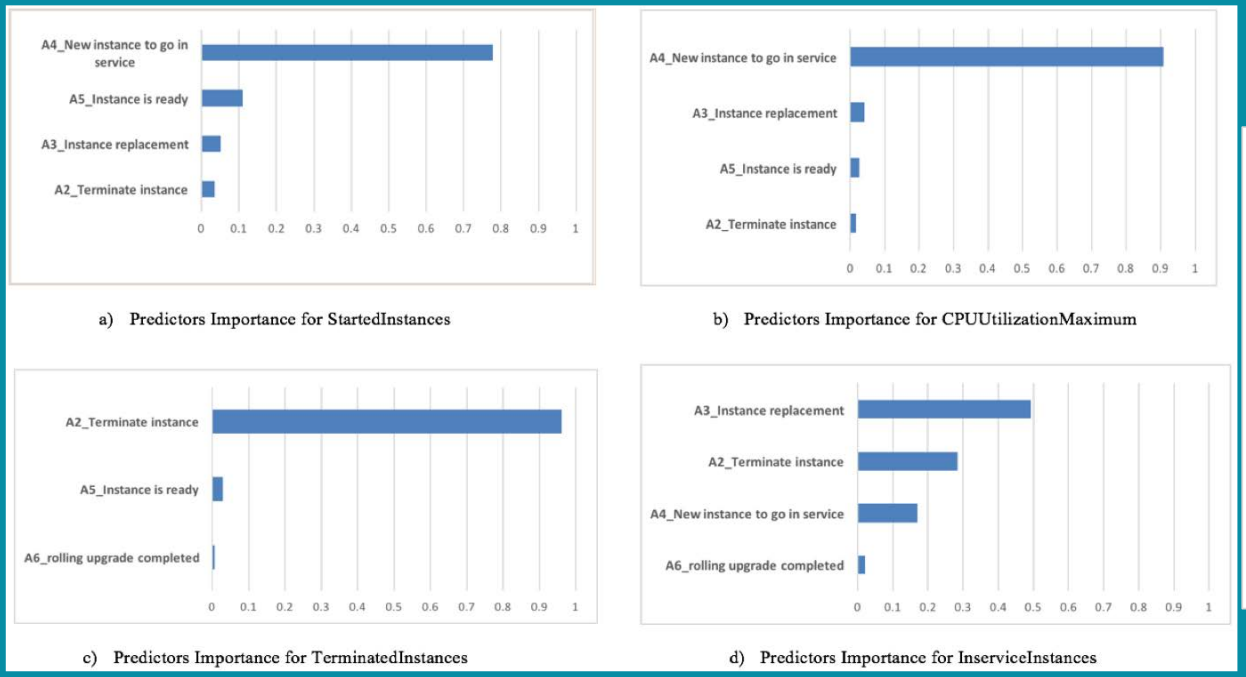

```
com.netflix.asgard.Task    2013-11-27_16:48:30 1401: {Ticket: null} {User: null} {Client:
localhost 0:0:0:0:0:0:0:1%0} {Region: ap-southeast-2} [Pushing ami-4f36aa75 into group
ASG-dsn for app ASG] Instance ASG on i-cdab74f1 is ready for use. 10 of 10 instance
relaunches done.
[2013-11-27 16:48:32,050] [Task:Pushing ami-4f36aa75 into group ASG-dsn for app ASG]
com.netflix.asgard.Task    2013-11-27_16:48:32 1401: {Ticket: null} {User: null} {Client:
localhost 0:0:0:0:0:0:0:1%0} {Region: ap-southeast-2} [Pushing ami-4f36aa75 into group
ASG-dsn for app ASG] Completed in 40m 2s.
[2013-07-12 16:07:32,753] [Task:Pushing ami-a105959b into group hadoopcluster for app
hadoopcluster] com.netflix.asgard.Task  2013-07-12_16:07:32 76: {Ticket: null} {User:
null} {Client: localhost 127.0.0.1} {Region: ap-southeast-2} [Pushing ami-a105959b into
group hadoopcluster for app hadoopcluster] Updating launch from
hadoopcluster-20130712152339 with ami-a105959b into hadoopcluster-20130712160732
[conformance:unclassified]
[2013-11-27 16:08:30,002] [Task:Pushing ami-4f36aa75 into group ASG-dsn for app ASG]
com.netflix.asgard.Task    2013-11-27_16:08:30 1401: {Ticket: null} {User: null} {Client:
localhost 0:0:0:0:0:0:0:1%0} {Region: ap-southeast-2} [Pushing ami-4f36aa75 into group
ASG-dsn for app ASG] Started on thread Task:Pushing ami-4f36aa75 into group ASG-dsn for
app ASG. [conformance:unfit]
[2013-11-27 16:08:30,637] [Task:Pushing ami-4f36aa75 into group ASG-dsn for app ASG]
com.netflix.asgard.Task    2013-11-27_16:08:30 1401: {Ticket: null} {User: null} {Client:
localhost 0:0:0:0:0:0:0:1%0} {Region: ap-southeast-2} [Pushing ami-4f36aa75 into group
ASG-dsn for app ASG] Updating launch from ASG-dsn-20501121075330 with ami-4f36aa75 into
ASG-dsn-20131127160830 [conformance:unfit]
[2013-11-27 16:08:30,639] [Task:Pushing ami-4f36aa75 into group ASG-dsn for app ASG]
com.netflix.asgard.Task    2013-11-27_16:08:30 1401: {Ticket: null} {User: null} {Client:
localhost 0:0:0:0:0:0:0:1%0} {Region: ap-southeast-2} [Pushing ami-4f36aa75 into group
ASG-dsn for app ASG] Create Launch Configuration 'ASG-dsn-20131127160830' with image
```

DEAKIN UNIVERSITY AUSTRALIA
Worldly


AMI
Launch
pending
Start
rebooting
Reboot
running
Stop
stopping
stopped
Terminate
shutting-down
terminated
Terminate
EBS-backed instances only


Rolling Upgrade in EC2
Update Auto Scaling Group
Sort Instances
Confirm Upgrade Spec
Remove & Deregister Old Instance from ELB
Terminate Old Instance
Wait for ASG to Start New Instance
Register New Instance with ELB


Monitoring tool/service
monitoring
EC2 Instances
Cloud operation
Start
Operation tool event-logs
Metrics
Collect monitoring metric data
Collect operation event-logs data
Extract direct and indirect cloud metrics
Derive measurable log-data metric
Log-Cloud metric mapping
Mapped Logs-Metircs Data
Logs Abstraction (Clustering)
Clustered Event-logs
Derive log-metric correlation model with regression analysis
Evaluate the Statistical Model
No
Statistical model Didn't fit the data.
Is the results valid?
Yes
1- Prediction formula
2- Main-factors affecting the target variable
Define assertion specification
End of Design/Training phase




a) Predictors Importance for StartedInstances



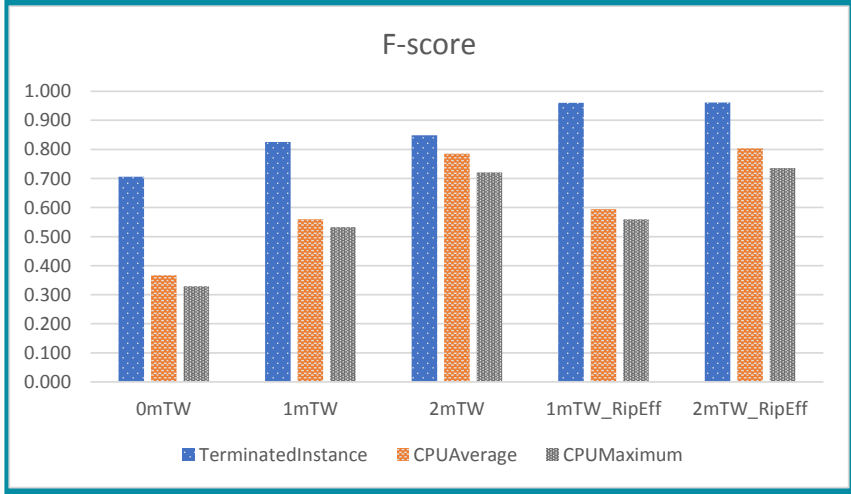

b) Predictors Importance for CPUUtilizationMaximum




c) Predictors Importance for TerminatedInstances




d) Predictors Importance for InserviceInstances

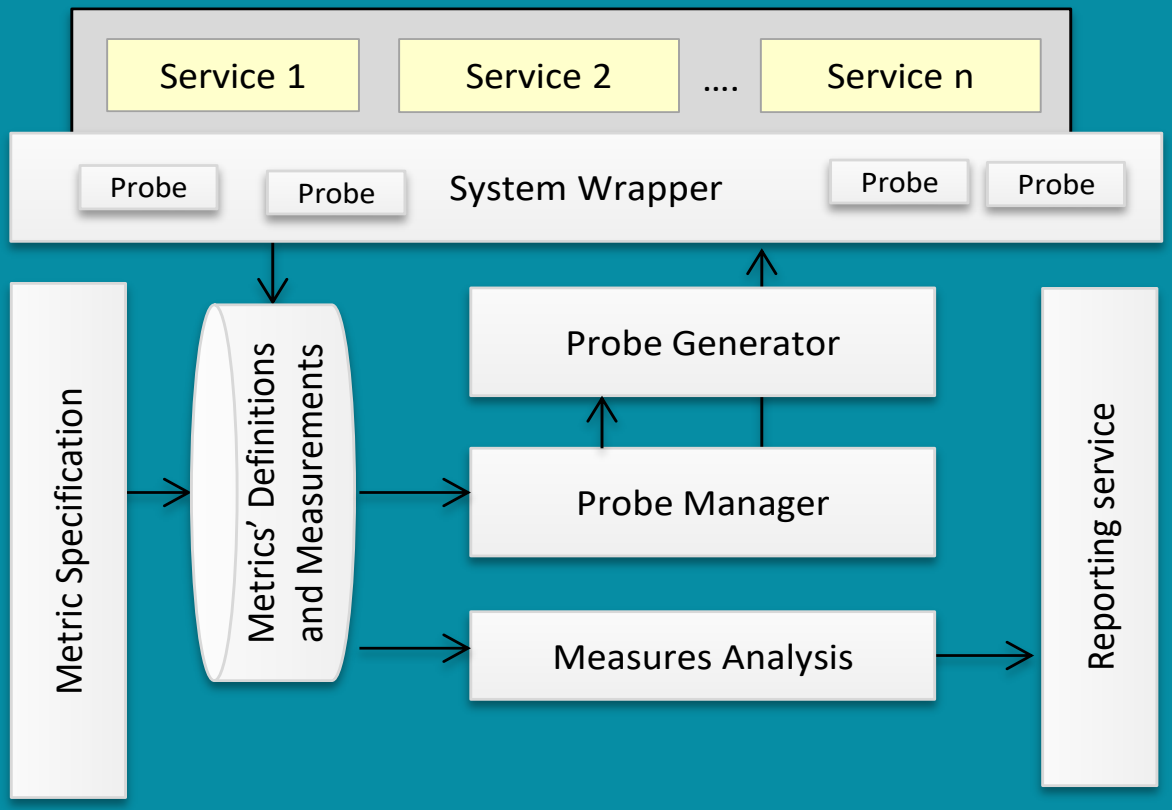


# TECHNIQUE #4 – MONITORING PROBE GENERATION

- How do we better monitor run-time metrics?
- Specify metrics and security constraints of interest – similar to vulnerability signatures
- Process application model to determine where to monitor
- Inject “probes” at run-time to monitor (using variety of techniques)
- Capture data, metrics
- Determine exceptions, mitigations
- Action mitigations…

# Example signatures of security metrics/properties in OCL

| Metric | Signature |
|---|---|
| **Information Disclosure** | context Method inv InfoDisclosure:<br>Let access : Request := self.Requests->last() in<br>Let authorized : Response :=<br>self.AuthorizationControl.Responses-> select(R\| R.IsValid = True AND access.UserID = R.UserID)->last() in IF (authorized) THEN true ENDIF |
| **Chinese Wall** | Let Subject := Classes->select(Name = 'Subj')->first() in<br>Let Obj: Class := Classes->select(Name = 'Object')->first()<br>Let mthdCall : Request := self.Requests->last() in<br>Let mthdReturn: Response := self.Responses->last() in<br>Let access : Request := self.Requests->last() in<br>IF (access.RequestTime > mthdCall.RequestTime and<br>access.RequestTime < mthdReturn.ResponseTime) THEN Not self.Conflictlist->exists(R\| R = access.Target) |
| **Restrict System Calls** | Let SystemCalls : Request := Classes->select(Name = 'SystemHandler')->first().Requests()->last() in<br>IF (SystemCalls <> null) THEN false ENDIF |
| **Separation of Duties** | Let xReq : Request:= Requests(Entity = 'MthdX') in<br>Let yReq : Request:= >Requests(Entity = 'MthdY') in<br>Let zReq : Request:= >Requests(Entity = 'MthdZ') in<br>IF (xReq.UserID = yReq.UserID and xReq.Target = yReq.Target Or xReq.UserID = zReq.UserID and zReq.Target = zReq.Target Or yReq.UserID = zReq.UserID and xReq.Target = yReq.Target) THEN false ENDIF |
| **Authenticated Requests** | context System inv AuthenticatedRequests:<br>self.AuthenticationControl.Requests->select()->count()/ self.Request->select()->count() |
| **Authentic Requests** | context System inv AuthenticRequests:<br>self.AuthenticationControl.Response->select(R \| R.IsValid = true)->count()/ self.AuthenticationControl.Request->select()->count() |
| **Last(10) Authz. Reqs** | context System inv Last10AuthzCtl:<br>self.AuthorizationControl.Requests->select()->Last(10) |
| **Top(10) admin Requests** | context System inv Top10AuthnCtl:<br>self.AuthenticationControl.Responses->select(R \| R.UserID = 'Admin')->count() |
| **Mean Time Between Unauthentic Request** | context System inv MTBUnauthenticRequests:<br>self.AuthenticationControl.Responses->select(R \| R.IsValid = false)>differences('Measurementtime')-> sum() / self.AuthenticationControl.Responses->select(R \| R.IsValid = false) )->count() |
| **Authenticated Requests Trend** | context System inv Authenticated RequestsTrend:<br>self.AuthenticatedRequests.Differences('AuthenticatedRequests')->sum() / self.AuthenticatedRequests-> count() |
| **MTBUR Over Systems** | context System inv MTBUROverSystems:<br>self.MTBUnauthenticRequests->sum()/ self.MTBUnauthenticRequests->count() |

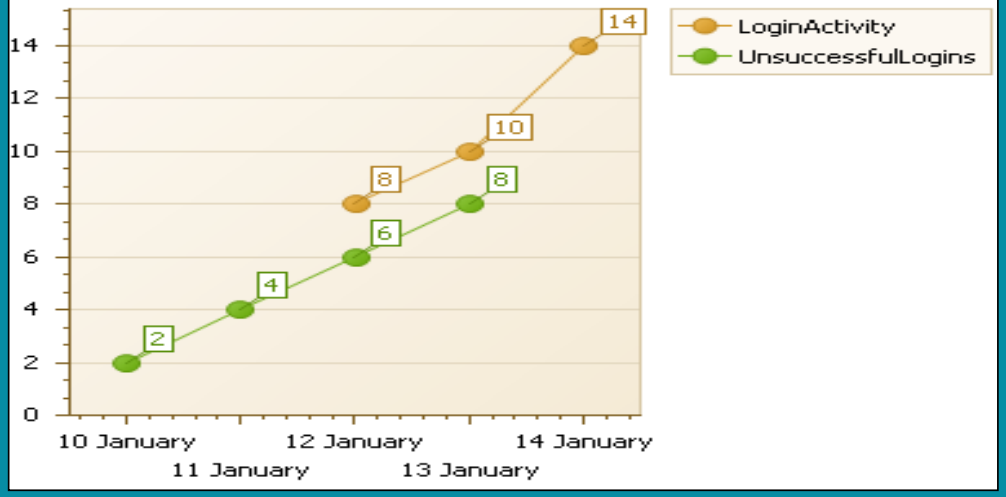

Service 1
Service 2
....
Service n
Probe
Probe
System Wrapper
Probe
Probe
Metric Specification
Metrics' Definitions and Measurements
Probe Generator
Probe Manager
Measures Analysis
Reporting service

Generated OCL Validation Code

```
public partial class AuthenticRequests {

    public static OCLReal AuthenticRequestsTest(IAgsiModelElement element) {
        OCLModelItem self = new OCLModelItem(element);
        OCLOrderedSet<OCLModelItem> sr_0 = self.GetModelNavigationMultiple("SecurityFn");
        OCLOrderedSet<OCLModelItem> return_1 = new OCLOrderedSet<OCLModelItem>();
        for (int i_2 = 0; (i_2 < sr_0.size()); i_2 = (i_2 + 1)) {
            OCLModelItem R = new OCLModelItem();
            R = sr_0[i_2];
            if ((((OCLString)(R.GetModelAttributeSimple("SecurityControlName"))).opEqual(new OCLString("AuthenticationControl")))) {
                return_1.including(R);
            }
        }
        OCLOrderedSet<OCLModelItem> sr_3 = return_1.first().GetModelNavigationMultiple("Responses");
        OCLOrderedSet<OCLModelItem> return_4 = new OCLOrderedSet<OCLModelItem>();
        for (int i_5 = 0; (i_5 < sr_3.size()); i_5 = (i_5 + 1)) {
            OCLModelItem D = new OCLModelItem();
            D = sr_3[i_5];
            if ((((OCLString)(D.GetModelAttributeSimple("IsValid"))).opEqual(new OCLBoolean(false)))) {
                return_4.including(D);
            }
        }
        OCLOrderedSet<OCLModelItem> sr_6 = self.GetModelNavigationMultiple("SecurityFn");
        OCLOrderedSet<OCLModelItem> return_7 = new OCLOrderedSet<OCLModelItem>();
        for (int i_8 = 0; (i_8 < sr_6.size()); i_8 = (i_8 + 1)) {
            OCLModelItem R = new OCLModelItem();
            R = sr_6[i_8];
            if ((((OCLString)(R.GetModelAttributeSimple("SecurityControlName"))).opEqual(new OCLString("AuthenticationControl")))) {
                return_7.including(R);
            }
        }
        return return_4.size().opDivide(return_7.first().GetModelNavigationMultiple("Requests").size());
    }
}
```

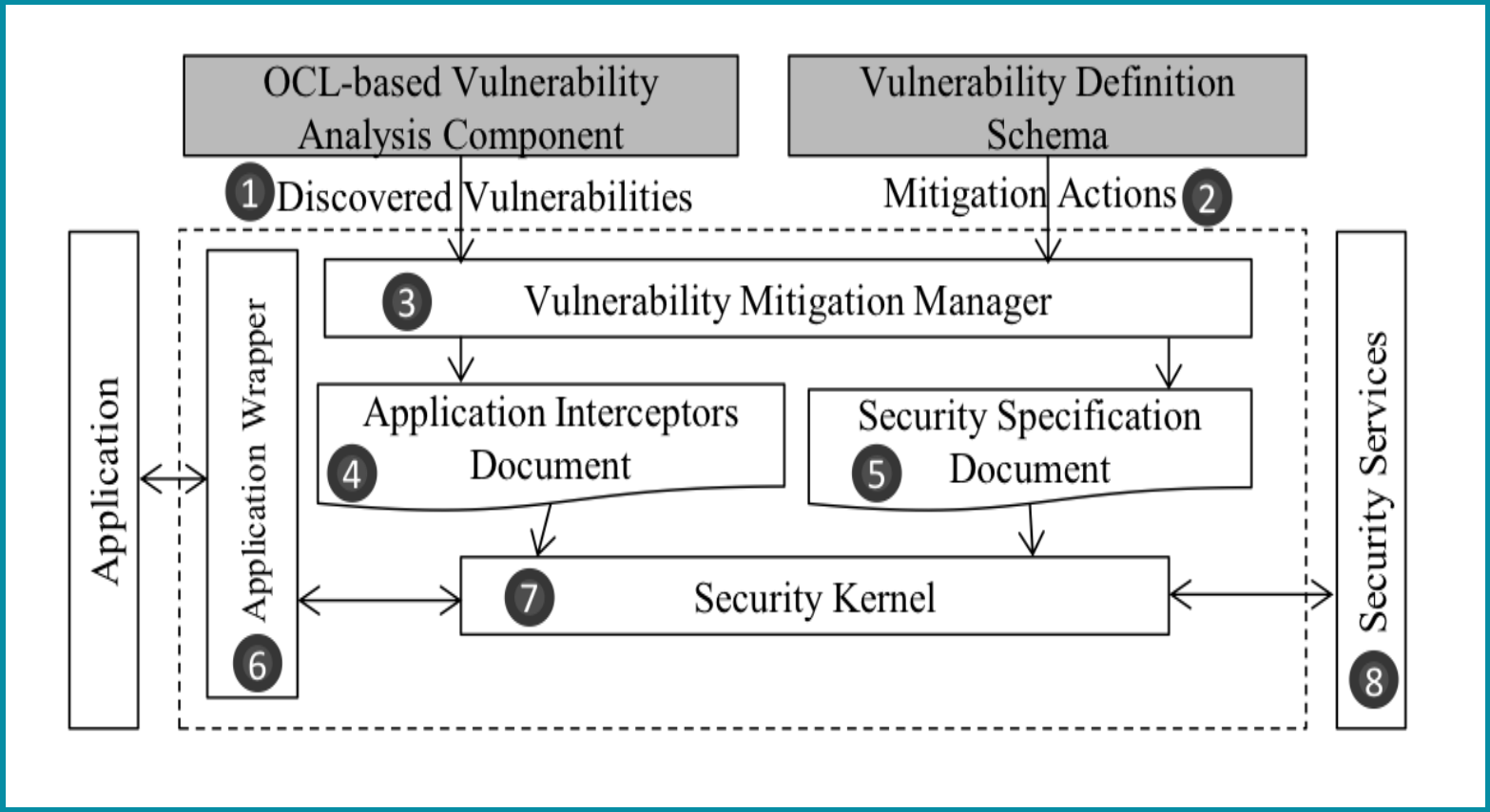
| Measurement Name | Measurement Date | Measurement Value |
|---|---|---|
| LoginActivity | 12/01/2011 | 8 |
| LoginActivity | 13/01/2011 | 10 |
| LoginActivity | 14/01/2011 | 14 |
| UnsuccessfulLogins | 10/01/2011 | 2 |
| UnsuccessfulLogins | 11/01/2011 | 4 |
| UnsuccessfulLogins | 12/01/2011 | 6 |
| UnsuccessfulLogins | 13/01/2011 | 8 |



# TECHNIQUE #5 – RUN-TIME MITIGATION

- Found vulnerability (statically or dynamically, at design-time or run-time) ; found anomaly – how fix / mitigate / raise alarm??
- Use one (or more) of previous techniques to identify security flaw / vulnerability / new attack scenario / anomalous measurement(s) / event(s) at run-time
- Identify feasible modification to application to address
- Update the application on-the-fly to address vulnerability / security flaw / counter attack scenario / mitigate for anomaly
- Validate that vulnerability etc has been addressed
- The beginnings of the notion of “self-securing software systems”

# TECHNIQUE #5 – RUN-TIME MITIGATION



# RE-ASPECTS GRAMMAR, SIGNATURES

| | | |
|---|---|---|
| Re-aspect Def | ::= | s:{Signature} a:{Action} d:{Advice} |
| Signature | ::= | st:Signature Type se: {Signature Expression} |
| Signature Type | ::= | code-snippet \| ocl-expression |
| Action | ::= | at:Action Type ac: {Action Condition} |
| Action Type | ::= | Delete \| Modify \| Replace \| Inject |
| Action Condition | ::= | ocl-expression |

**Figure 7: Re-aspect Grammar**

```
if( Request.Cookies["Loggedin"] != true ) {
    if(  !AuthenticateUser(Request.Params["username"],
                           Request.Params["password"] ) );
        throw new Exception("Invalid user");
}
DoAdministration();
```

**Figure 3: Case 2: code vulnerable to authentication bypass, to replace**

```
if( !AuthenticateUser( Request.Params["username"],
                       Request.Params["password"] ) )
    throw new Exception("Invalid user");
if( !AuthorizeUser( Thread.CurrentPrincipal,
        (new StakeFrame()).GetMethod().Name,
        (new StakeFrame()).GetMethod().GetParameters() ) )
    throw new Exception("User is not auhorized");
updateCustomerBalance(Request.QueryString["cID"], nBalance);
```

**Figure 6: Case 4: code vulnerable to improper authorization, to inject**

```
bool updateCustomerBalance(string custID, decimal nBalance) {
    if(!AuthenitcateUser( username, password)) return false;
    if(!AuthorzUser(username, "updateCustBalance")) return false;
    LogTrx(username, dateTime.Now, "updateCustomerBalance");
    Customer customer = Customers.getCustomerByID(custID);
    customer.Balance = nBalance;
    Customers.SaveChanges();
    LogTrx(username, dateTime.Now, "updateCustBalance done");
}
```

**Figure 2: Case 1: code with old security functions, we want to leave out**

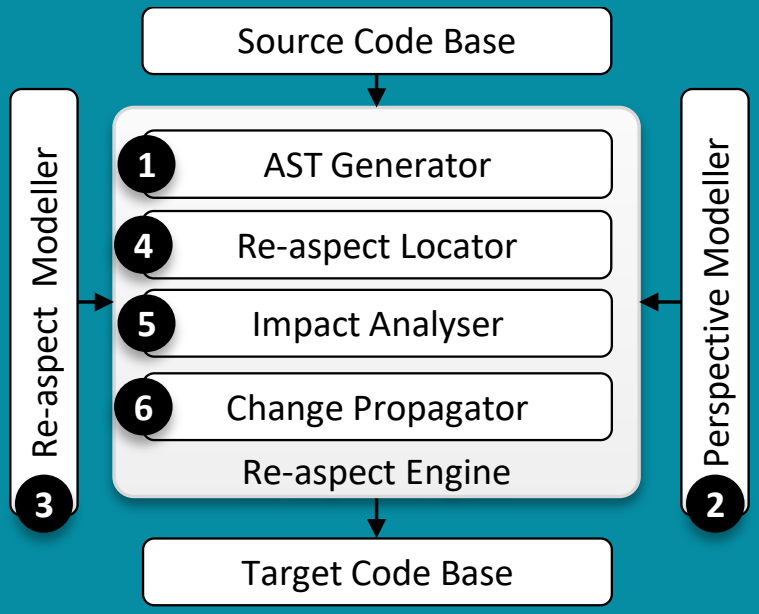
```
Inputsanitizer( (new StakeFrame()).GetMethod().GetParameters() );
string query = "SELECT *  FROM USERS WHERE UserID = '"
+ EncodeForSQL(username)  + "' AND password = '"
+ EncodeForSQL(password)  + "'";
```

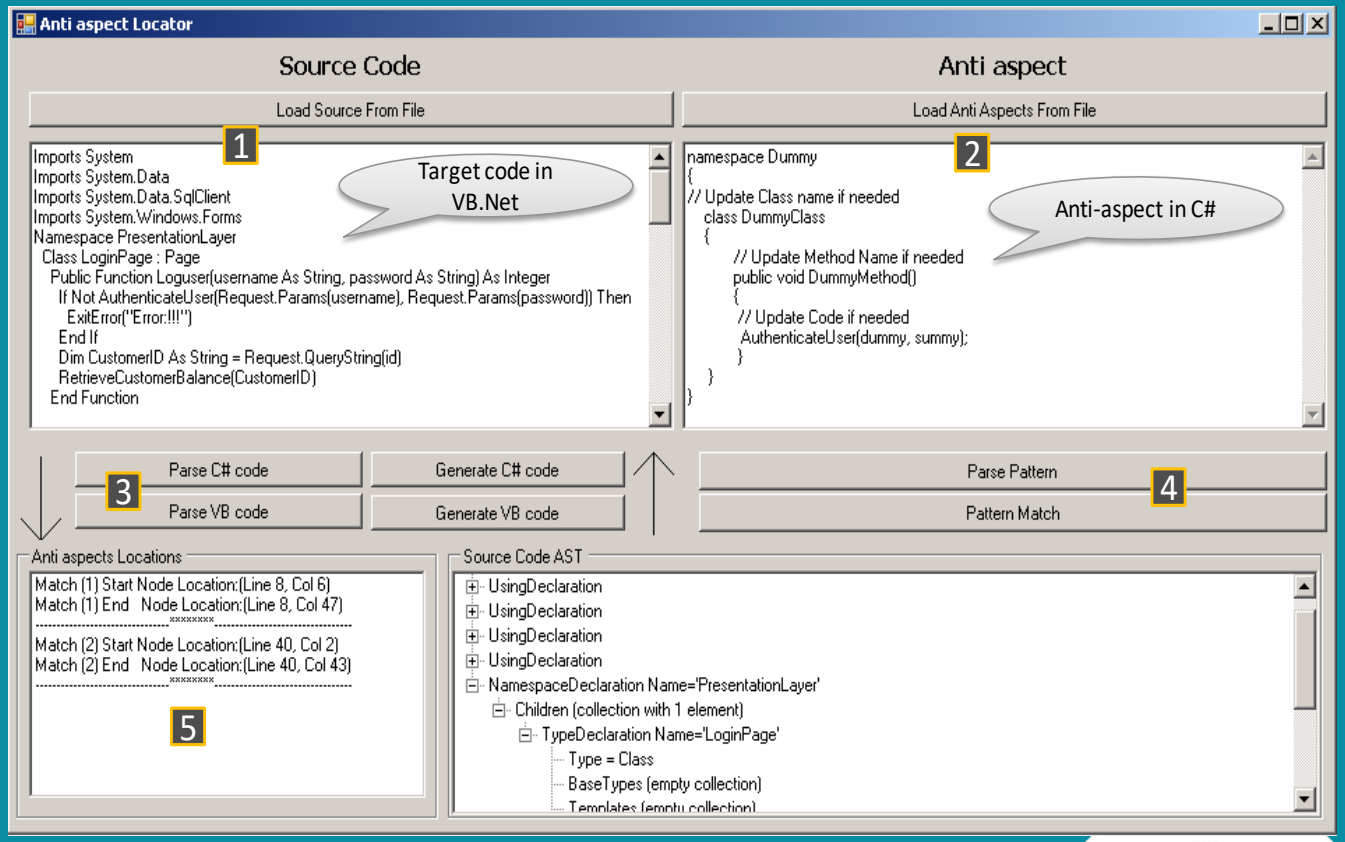
**Figure 5: Case 3b: Code vulnerable to SQL injection, to modify**



DEAKIN UNIVERSITY AUSTRALIA Worldly

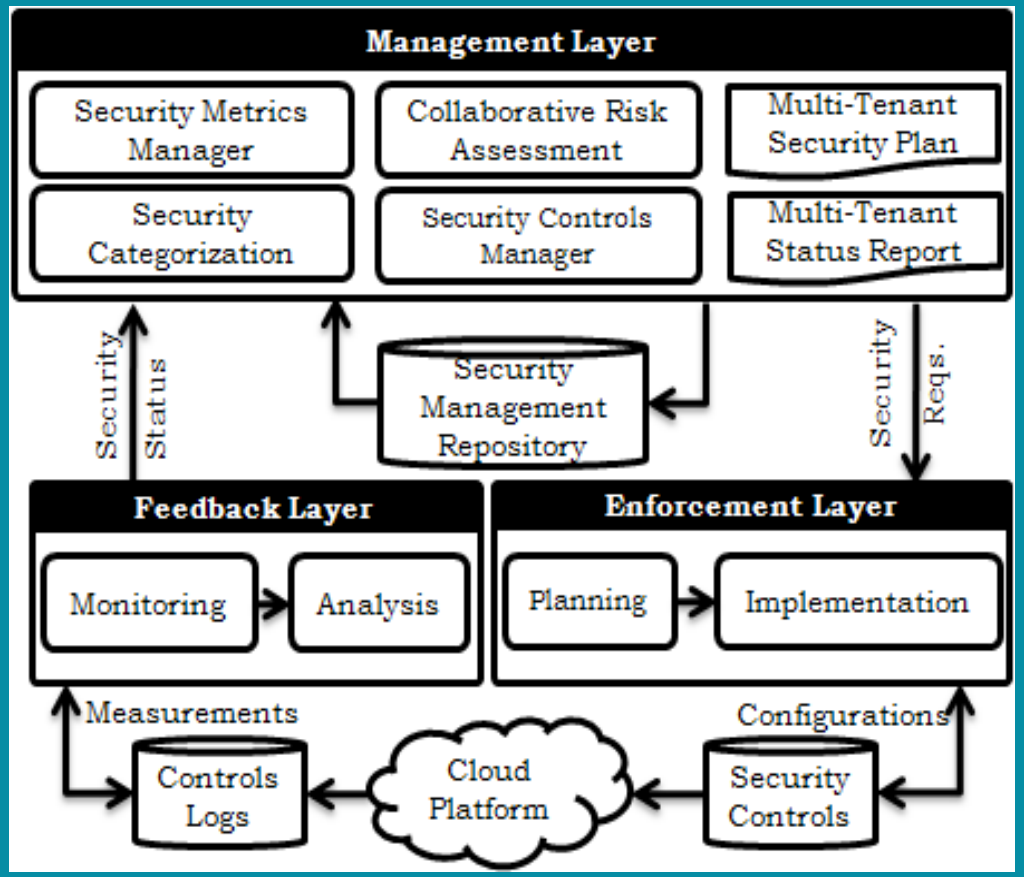
# SMART TOOL


Source Code Base
Re-aspect Modeller
3
1
AST Generator
4
Re-aspect Locator
5
Impact Analyser
6
Change Propagator
Re-aspect Engine
Perspective Modeller
2
Target Code Base


Anti aspect Locator
Source Code
Anti aspect
Load Source From File
Load Anti Aspects From File
1
2
Target code in VB.Net
Anti-aspect in C#
Imports System
Imports System.Data
Imports System.Data.SqlClient
Imports System.Windows.Forms
Namespace PresentationLayer
Class LoginPage : Page
Public Function Loguser(username As String, password As String) As Integer
If Not AuthenticateUser(Request.Params(username), Request.Params(password)) Then
ExitError("Error:!!!")
End If
Dim CustomerID As String = Request.QueryString(id)
RetrieveCustomerBalance(CustomerID)
End Function
namespace Dummy
{
// Update Class name if needed
class DummyClass
{
// Update Method Name if needed
public void DummyMethod()
{
// Update Code if needed
AuthenticateUser(dummy, summy);
}
}
}
Parse C# code
Generate C# code
Parse VB code
Generate VB code
3
Parse Pattern
Pattern Match
4
Anti aspects Locations
Match (1) Start Node Location:(Line 8, Col 6)
Match (1) End Node Location:(Line 8, Col 47)
Match (2) Start Node Location:(Line 40, Col 2)
Match (2) End Node Location:(Line 40, Col 43)
5
Source Code AST
UsingDeclaration
UsingDeclaration
UsingDeclaration
UsingDeclaration
NamespaceDeclaration Name='PresentationLayer'
Children (collection with 1 element)
TypeDeclaration Name='LoginPage'
Type = Class
BaseTypes (empty collection)
Templates (empty collection)

# TECHNIQUE #6 – TENANT-ORIENTED SECURITY CONTROLS

- Cloud applications using SaaS model typically have multiple tenants sharing same software / platform / infrastructure
- But – different tenants may have different security requirements
- How support this – at SaaS/PaaS or even IaaS levels?!
- Different tenants specifying security requirements – user model
- Realising different tenant security requirements on same platform




| # | Ctl Family | Ctl No. | Enhancement | Ctl Name | Control Status |
|---|---|---|---|---|---|
| Edit Delete | AC- | 14 | 1 | | Missing |
| Edit Delete | AC- | 17 | 1 | Authenitcator | Available |
| Edit Delete | AC- | 17 | 1 | SwinAntiVirus | Duplicate |
| Edit Delete | AC- | 17 | 2 | Authenitcator | Available |
| Edit Delete | AC- | 17 | 2 | SwinAntiVirus | Duplicate |

The Security Management plan for the service Galactic EPP Service

| # | Registration Date | Registration (Mths) | Security Categorization |
|---|---|---|---|
| | 1/01/2011 | 24 | Low |

| Vulnerability Name | Vulnerability Description |
|---|---|
| CVE-2005-0413 | Multiple SQL injection vulnerabilities in MyPHP Forum 1.0 allow remote attackers to execute arbitra |
| CVE-2005-2471 | pstopnm in netpbm does not properly use the "-dSAFER" option when calling Ghostscript to convert a |
| CVE-2005-4195 | Multiple SQL injection vulnerabilities in Scout Portal Toolkit (SPT) 1.3.1 and earlier allow remote |

| Threat Name | Threat Description | Threat Source |
|---|---|---|
| DenialSrv | Denial of service | Attacker |
| InfoCopy | Copy of information at storage | Internal |
| InfoMod | Modification of information while being transfered | Attacker |
| MaintMod | Modification of data being processed | Malware |

| Risk Name | Risk Probability | Confidentiality Impact | Availability Impact | Integrity Impact | Risk Level |
|---|---|---|---|---|---|
| DOS | 0.7 | Low | High | Low | Medium |

| Control Name | Control Description | Control Baseline | Control Type | Control Family |
|---|---|---|---|---|
| Authenticator | an authenitcation security control | Low | Specific | Access Control |
| SwinAntiVirus | an antivirus security solution | Low | Common | System and Information Integrity |
| SwinIPS | an intrusion prevention system | Low | CommonControl | System and Information Integrity |

| Measurement Name | Measurement Description | Frequency | Measurement Steps | Security Control |
|---|---|---|---|---|
| LoginActivity | Identify the user login rates | 48 | count(logstatus) | Authenticator |

# ALL IS NOT AS IT MAY SEEM…

- Can compare systems in the same domain – but appearances can be (very) deceiving…
- Vulnerability Counts vs Metrics vs meaning
  - need to compare like with like
  - Criticality of the issue vs simple occurrences
  - System scale makes a large difference
- Just one critical weakness can cause whole system to be compromised under attack; lots of minor weaknesses may be tolerable
- Its rather slow to analyse many of these => non-real time
- Change to environment / co-deployed services/applications => changes to measures / counts…
- Run-time vulnerability analysis still emerging area

# CURRENT / FUTURE WORK

- Further formalisation of the OWSAP and CAPEC databases of security vulnerabilities (IMO one of the real contributions we have undersold…)
- Apply deep learning to static, dynamic vulnerability detection vs rule-based (DIGGER, SMART) and statistical-based (log analysis) approaches – have a group of leading experts @ Deakin on this ☺
- Implies have good training set - but…
- Implies have good vector model for input to the RNN-based learnerc- but...
- Supporting tenants to specify their security requirements is... Really hard!
- Zero-day threat detection at IaaS level extremely hard – but working on how to apply to IoT security analysis and mitigation

# Questions ?


DEAKIN
UNIVERSITY AUSTRALIA
Worldly

# REFERENCES


- Almorsy, M., Grundy, J.C., Ibrahim, A., Adaptive Software Security, Chapter 5 in Managing trade-offs in adaptable software architectures, I. Mistrik, J. Grundy, B. Schmerl, R. Kazman, N. Ali (Eds), Morgan Kaufmann, January 2016.
- Almorsy, M., Grundy, J.C. and Ibrahim, A. Improving Tenants' Trust In SaaS Applications Using Dynamic Security Monitors, In 2015 International Conference on Engineering Complex Computing Systems (ICECCS 2015), Gold Coast, Australia, 9-12 December, IEEE
- Almorsy, M., Grundy, J.C., Ibrahim, A., Adaptable, Model-driven Security Engineering for SaaS Cloud-based Applications, Automated Software Engineering, vol. 21, no. 2, April 2014, Springer.
- 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.
  Almorsy, M., Grundy, J.C. and Ibrahim, A., Supporting Automated Software Re-Engineering Using "Re-Aspects", 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., Almorsy, M., Operating System Kernel Data Disambiguation to Support Security Analysis, 2012 International Conference on Network and System Security (NSS 2012), Fujian, China, Nov 21-23 2012, LNCS, Springer.
- Almorsy, M., Grundy, J.C. and Imbrahim, A. Collaboration-Based Cloud Computing Security Management Framework, In Proceedings of 2011 IEEE International Conference on Cloud Computing (CLOUD 2011), Washington DC, USA on 4 July – 9 July, 2011, IEEE.
- Imbrahim, A., Hamlyn-Harris J., Grundy, J.C. and Almorsy, M., CloudSec: A Security Monitoring Appliance for Virtual Machines in the IaaS Cloud Model, In Proceedings of the 5th International Conference on Network and System Security (NSS 2011), Milan, Italy, September 5-7 2011, IEEE 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. and Almorsy, M., DIGGER: Identifying OS Kernel Objects for Run-time Security Analysis, International Journal on Internet and Distributed Computing Systems, vol 3, no. 1, January 2013, pp 184-194.
- Almorsy, M. and Grundy, J.C. SecDSVL: A Domain-Specific Visual Language To Support Enterprise Security Modelling, 2014 Australasian Conference on Software Engineering (ASWEC 2014), Sydney, Australia, April 2014, IEEE CS Press.
- Almorsy, M., Grundy, J.C., Ibrahim, A., SMURF: Supporting Multi-tenancy Using Re-Aspects Framework, 17th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2012), Paris, France, July 2012, IEEE CS Press.