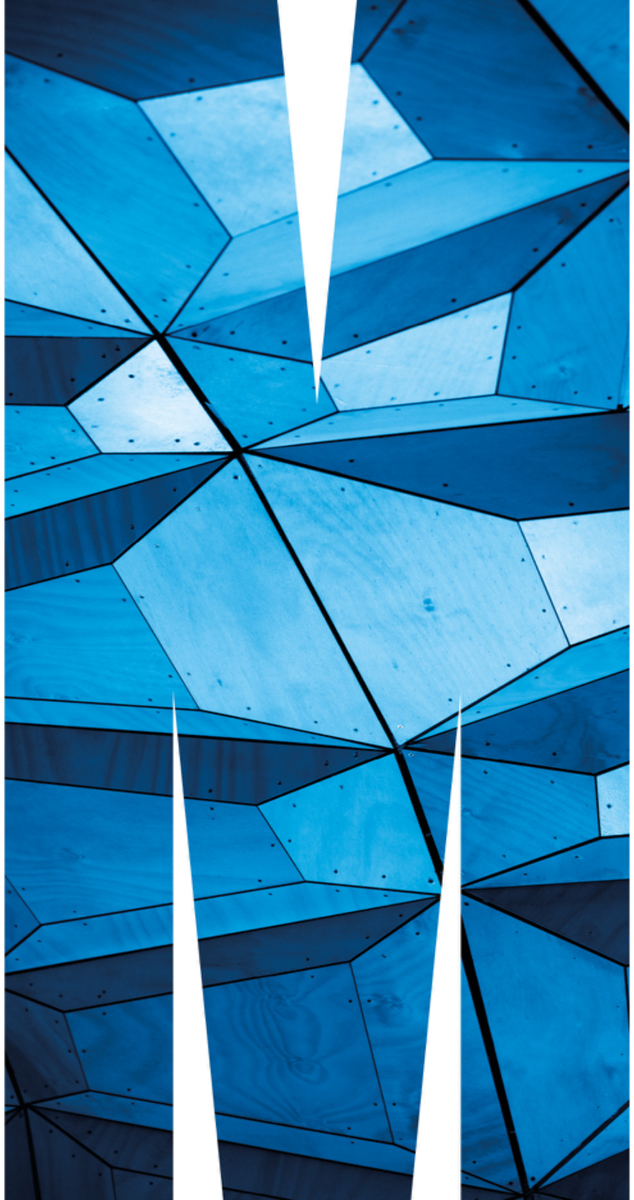
MONASH
INFORMATION
TECHNOLOGY

Towards self-securing software systems

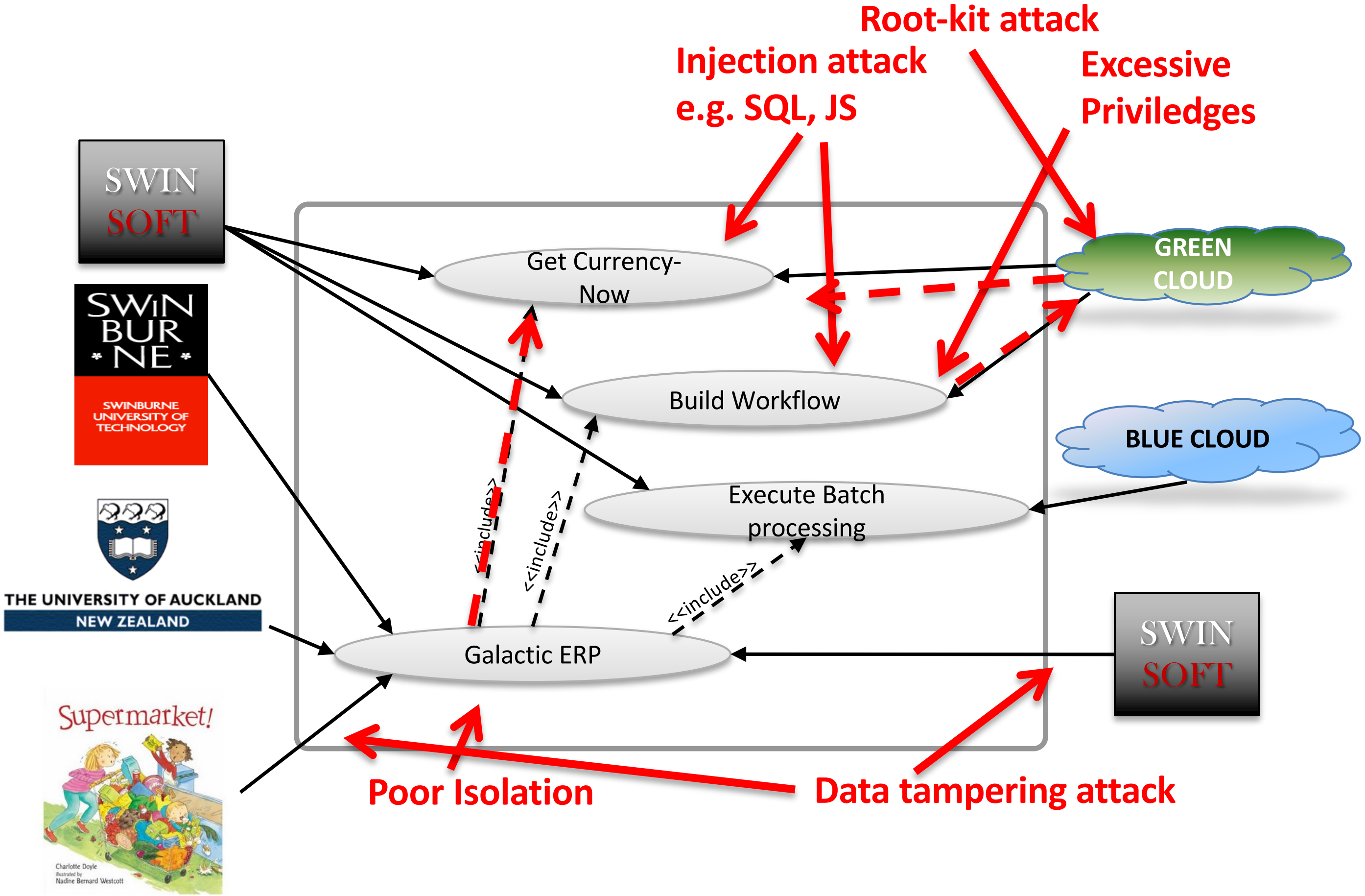
Professor John Grundy
Dr Mohamed Abdelrazek – Deakin
Dr Amani Ibrahim - Deakin
Mostafa Farshchi



Outline

- Motivating example
- Some (partial) solutions we have been working on:
  - Static vulnerability analysis
  - Log / metric correlation analysis (dynamic analysis)
  - Run-time cloud monitoring via generated probes (static & dynamic)
  - Mitigation via run-time software update (models @ run-time approach)
- Future directions…

Motivation


Root-kit attack
Injection attack
e.g. SQL, JS
Excessive
Priviledges
SWIN
SOFT
SWIN
BUR
NE
SWINBURNE
UNIVERSITY OF
TECHNOLOGY
THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Supermarket!
Get Currency-Now
Build Workflow
Execute Batch processing
Galactic ERP
<<include>>
<<include>>
<<include>>
GREEN CLOUD
BLUE CLOUD
SWIN
SOFT
Poor Isolation
Data tampering attack

“Self-securing” Software Systems

- Some key challenges:
  - When engineer cloud applications, don’t know what other apps be deployed with, hardware deployed on, networks etc
  - Stakeholder requirements change esp multi-tenant cloud apps
  - New threats continually emerging
  - Design-time fixing / re-deploying too slow, leaves system vulnerable

- Idea is to have the software itself:
  - Identify emergent threats - even as its environment changes
  - Identify mitigations to the threats
  - Self-adapt the application(s) while in use to counter the threat

Technique #1 – 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

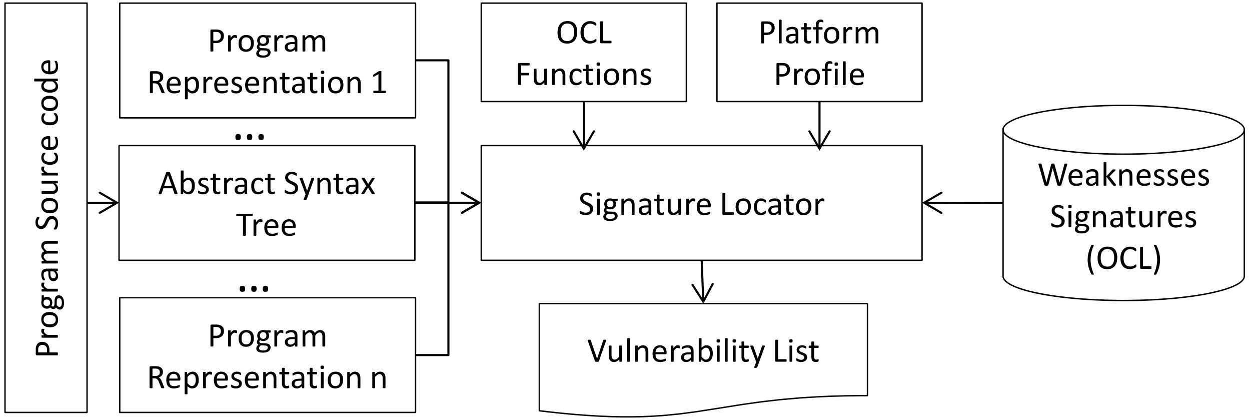
```
if( !AuthenticateUser( Request.Params["username"],
                            Request.Params["password"] ) )
             throw new Exception("Invalid user");
updateCustomerBalance(Request.QueryString["custID"], nBalance);
```

Figure 4. A code snippet vulnerable to improper authz

Formal vulnerability signatures

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)

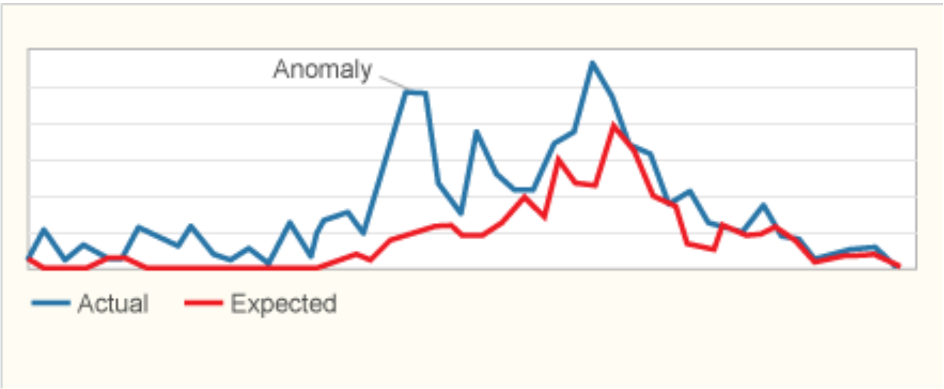
SMART (Static) Analyser

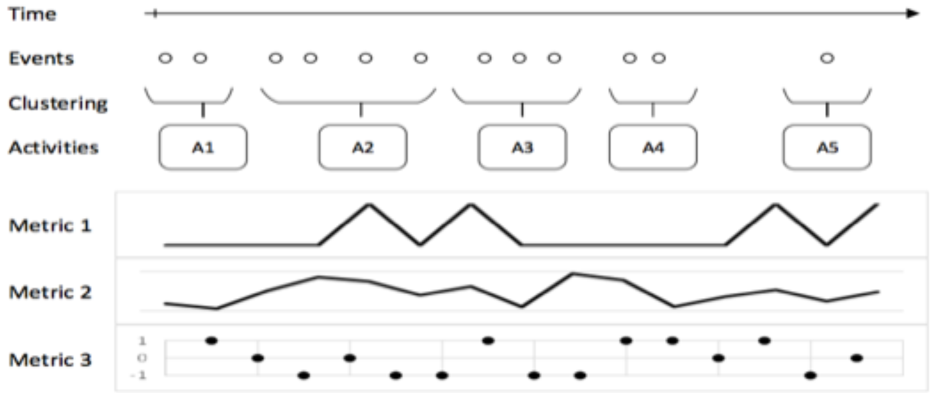


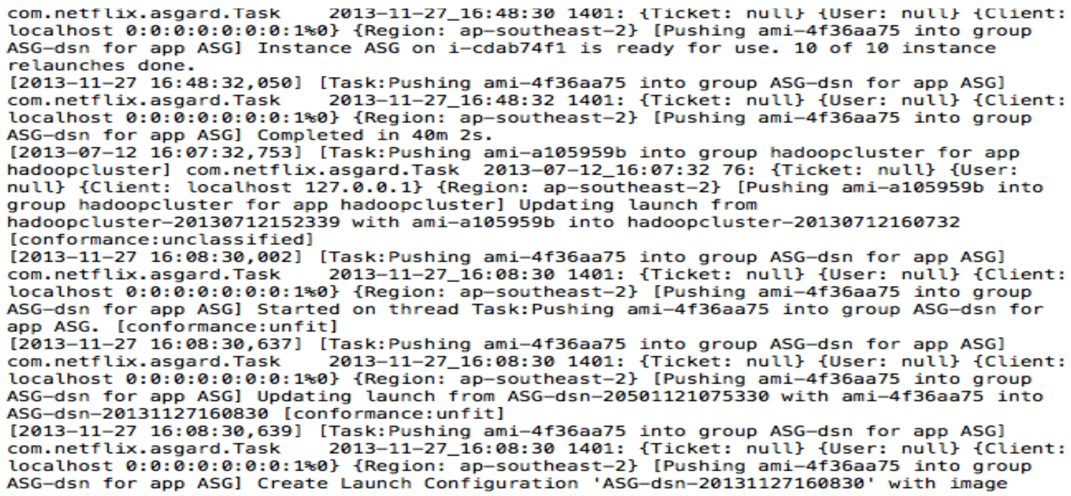
Technique #2 – log file/cloud PaaS metric analysis (dynamic analysis)

- 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

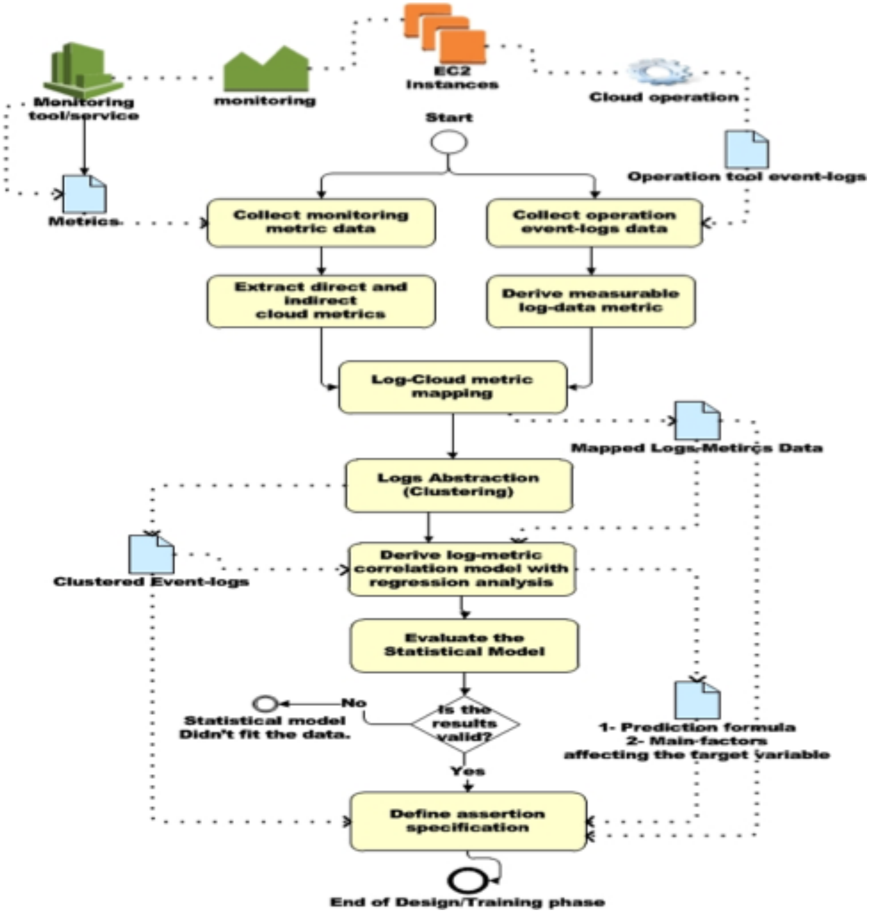




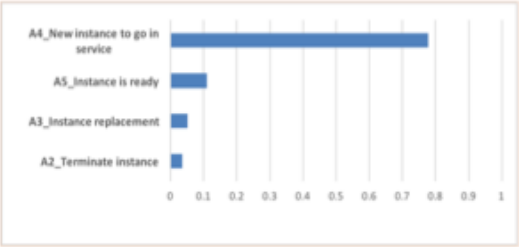


```
com.netflix.asgard.Task    2013-11-27_16:48:30 1401: {Ticket: null} {User: null} {Client:
localhost 0:0:0:0:0:0:0:1%0} {Region: ap-southeast-2} [Pushing ami-4f36aa75 into group
ASG-dsn for app ASG] Instance ASG on i-cdab74f1 is ready for use. 10 of 10 instance
relaunches done.
[2013-11-27 16:48:32,050] [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
```

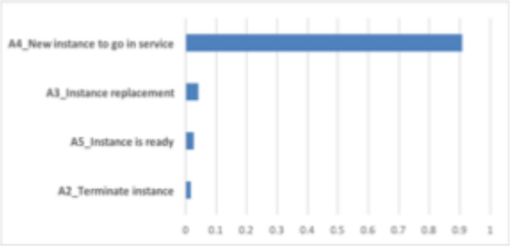
Process…


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

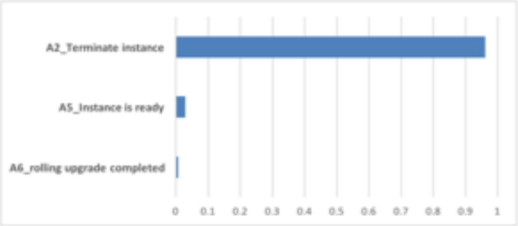
Correlation analysis



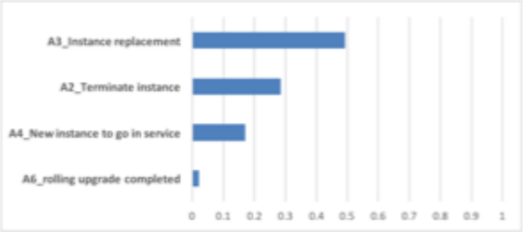
a) Predictors Importance for StartedInstances



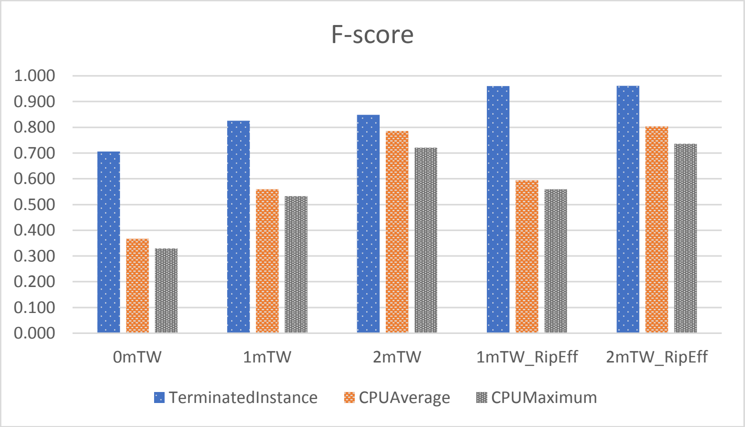
b) Predictors Importance for CPUUtilizationMaximum



c) Predictors Importance for TerminatedInstances



d) Predictors Importance for InserviceInstances



Technique #3 – monitoring probe generation

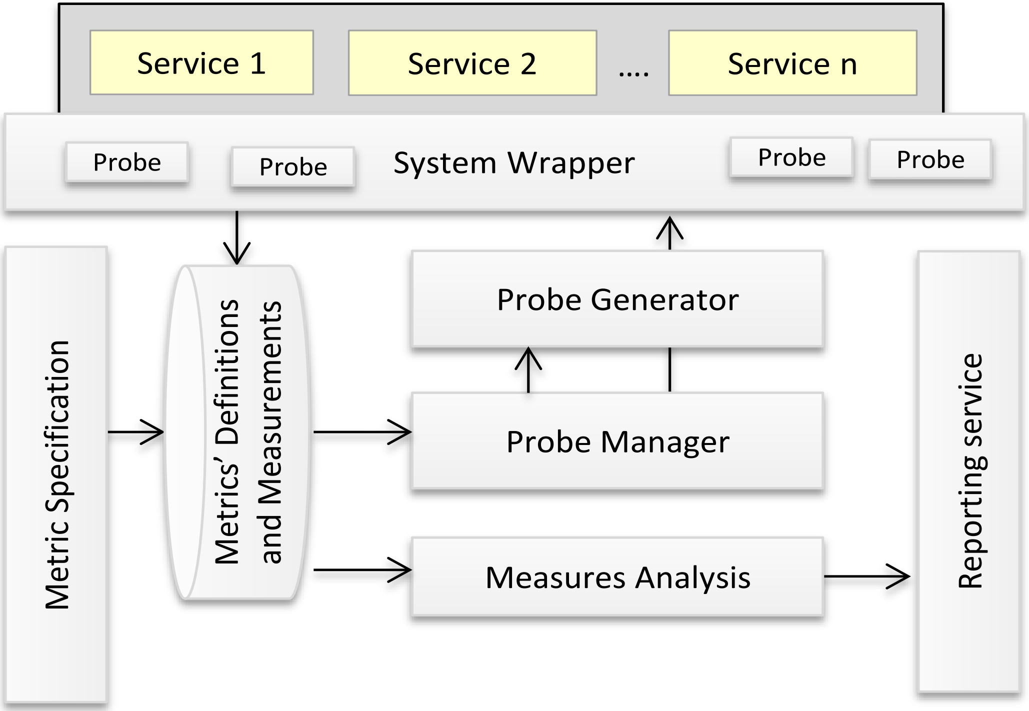
- How do we better monitor run-time metrics?
- Specify metrics and security constraints of interest – similar to vulnerability signatures
- Process application model to determine where to monitor
- Inject “probes” at run-time to monitor (using variety of techniques)
- Capture data, metrics
- Determine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

Example signatures of security metrics/properties in OCL

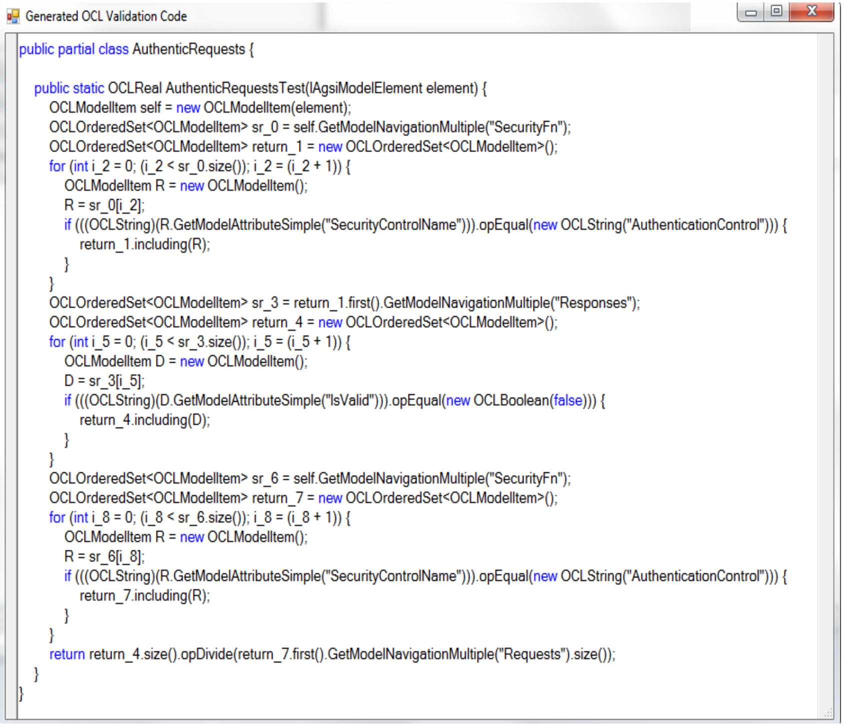
Metric	Signature
Information Disclosure	context Method inv InfoDisclosure: Let access : Request := self.Requests->last() in Let authorized : Response := self.AuthorizationControl.Responses-> select(R| R.IsValid = True AND access.UserID = R.UserID)->last() in IF (authorized) THEN true ENDIF
Chinese Wall	Let Subject := Classes->select(Name = 'Subj')->first() in Let Obj: Class := Classes->select(Name = 'Object')->first() Let mthdCall : Request := self.Requests->last() in Let mthdReturn: Response := self.Responses->last() in Let access : Request := self.Requests->last() in IF (access.RequestTime > mthdCall.RequestTime and 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 IF (SystemCalls <> null) THEN false ENDIF
Separation of Duties	Let xReq : Request:= Requests(Entity = 'MthdX') in Let yReq : Request:= >Requests(Entity = 'MthdY') in Let zReq : Request:= >Requests(Entity = 'MthdZ') in 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: self.AuthenticationControl.Requests->select()->count()/ self.Request->select()->count()
Authentic Requests	context System inv AuthenticRequests: self.AuthenticationControl.Response->select(R | R.IsValid = true)->count()/ self.AuthenticationControl.Request->select()->count()
Last(10) Authz. Reqs	context System inv Last10AuthzCtl: self.AuthorizationControl.Requests->select()->Last(10)
Top(10) admin Requests	context System inv Top10AuthnCtl: self.AuthenticationControl.Responses->select(R | R.UserID = ‘Admin’)->count()
Mean Time Between Unauthentic Request	context System inv MTBUnauthenticRequests: 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: self.AuthenticatedRequests.Differences(‘AuthenticatedRequests’)->sum() / self.AuthenticatedRequests-> count()
MTBUR Over Systems	context System inv MTBUROverSystems: self.MTBUnauthenticRequests->sum()/ self.MTBUnauthenticRequests->count()

MONASH University

Process

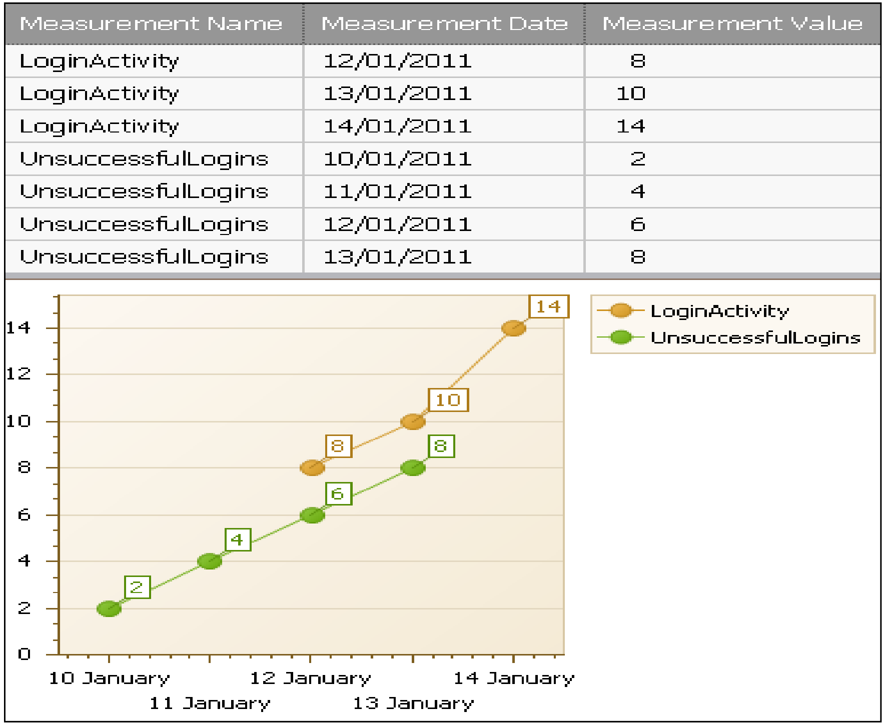

Service 1
Service 2
....
Service n
Probe
Probe
System Wrapper
Probe
Probe
Metric Specification
Metrics' Definitions and Measurements
Probe Generator
Probe Manager
Measures Analysis
Reporting service

Results



```
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());
   }
}
```

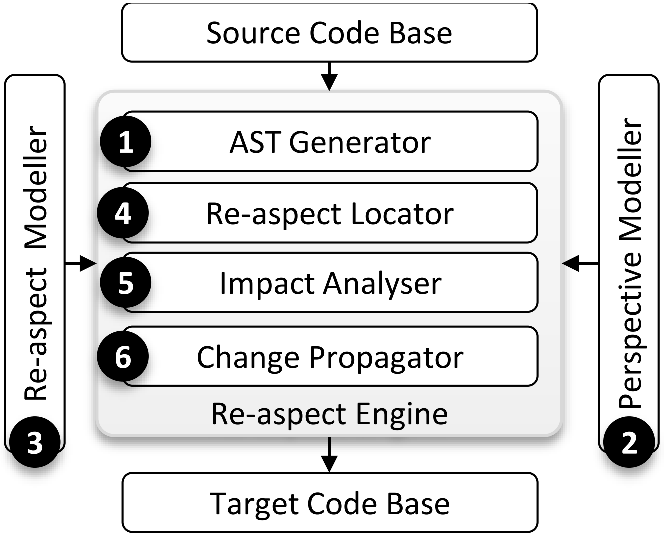


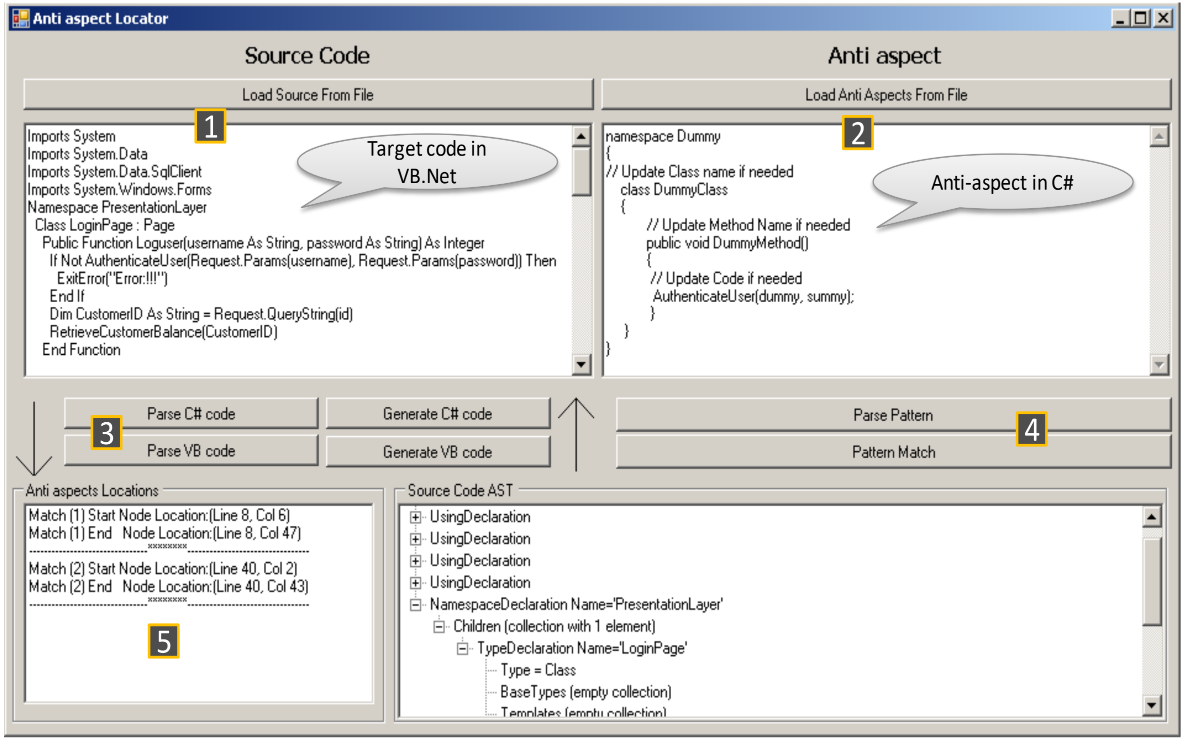
Measurement Name	Measurement Date	Measurement Value
LoginActivity	12/01/2011	8
LoginActivity	13/01/2011	10
LoginActivity	14/01/2011	14
UnsuccessfulLogins	10/01/2011	2
UnsuccessfulLogins	11/01/2011	4
UnsuccessfulLogins	12/01/2011	6
UnsuccessfulLogins	13/01/2011	8

Technique #4 – 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”…

SMART Tool – code updater





```
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
```

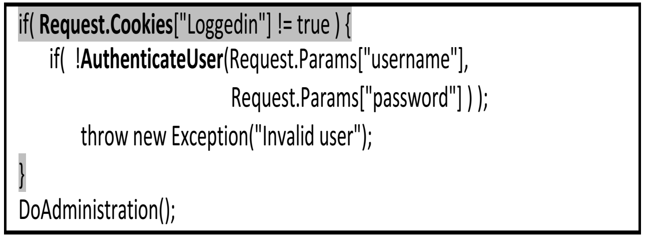
```
namespace Dummy
{
// Update Class name if needed
   class DummyClass
   {
        // Update Method Name if needed
        public void DummyMethod()
        {
         // Update Code if needed
          AuthenticateUser(dummy, summy);
         }
     }
}
```

```
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)
----------------------------********----------------------------
```

```
⊞ UsingDeclaration
⊞ UsingDeclaration
⊞ UsingDeclaration
⊞ UsingDeclaration
⊟ NamespaceDeclaration Name='PresentationLayer'
   ⊟ Children (collection with 1 element)
      ⊟ TypeDeclaration Name='LoginPage'
          Type = Class
          BaseTypes (empty collection)
          Templates (empty collection)
```

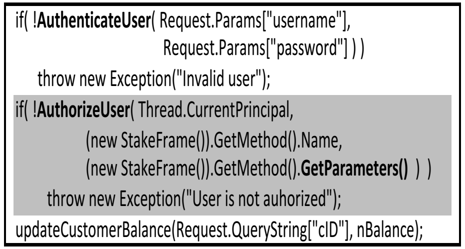
MONASH University

Fix ups of vulnerable code



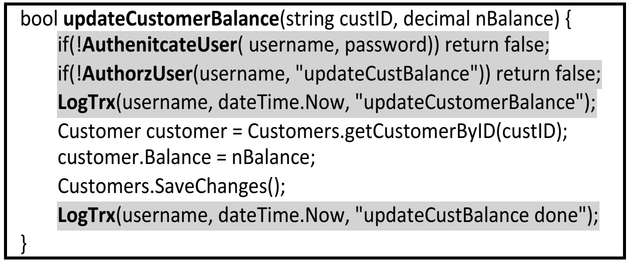
```
if( Request.Cookies["Loggedin"] != true ) {
     if(  !AuthenticateUser(Request.Params["username"],
                            Request.Params["password"] ) );
          throw new Exception("Invalid user");
}
DoAdministration();
```

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



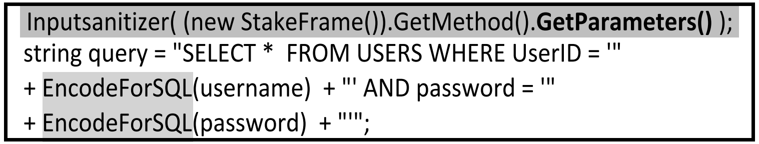
```
if( !AuthenticateUser( Request.Params["username"],
                       Request.Params["password"] ) )
     throw new Exception("Invalid user");
if( !AuthorizeUser( Thread.CurrentPrincipal,
            (new StakeFrame()).GetMethod().Name,
            (new StakeFrame()).GetMethod().GetParameters()  )  )
      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");
}
```

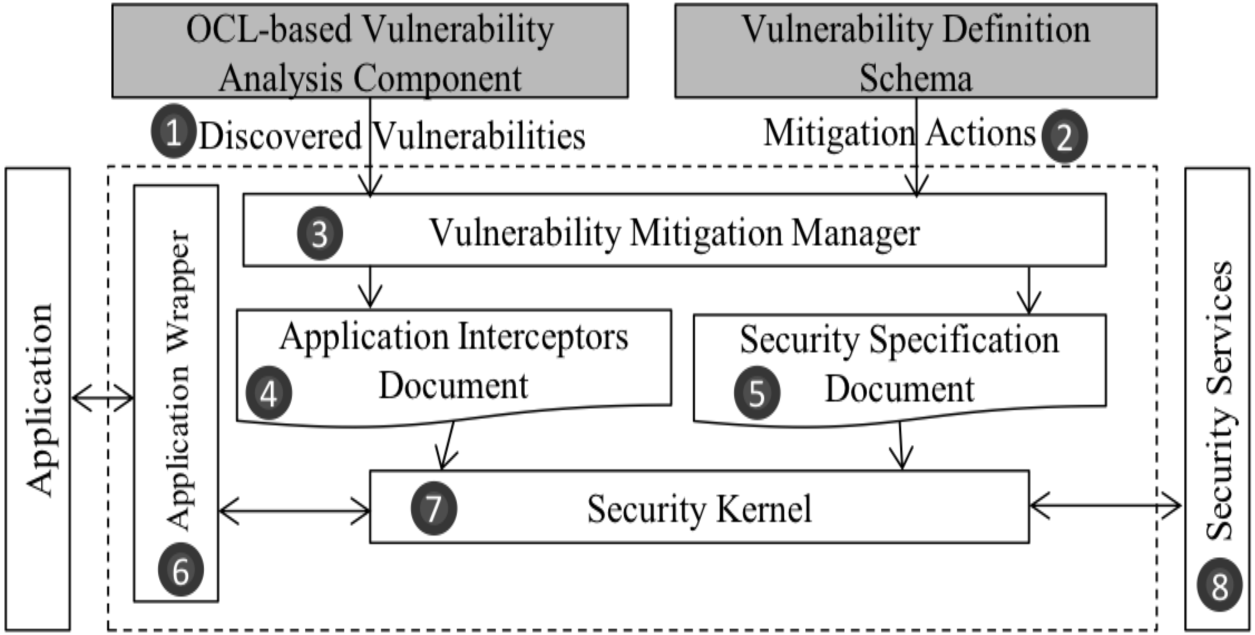
Figure 2: Case 1: code with old security functions, we want to leave out



```
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

Run-time mitigation architecture via reconfiguration


OCL-based Vulnerability Analysis Component
Vulnerability Definition Schema
1 Discovered Vulnerabilities
Mitigation Actions 2
3 Vulnerability Mitigation Manager
Application Interceptors Document 4
Security Specification Document 5
Application
Application Wrapper 6
7 Security Kernel
Security Services 8

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…

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.