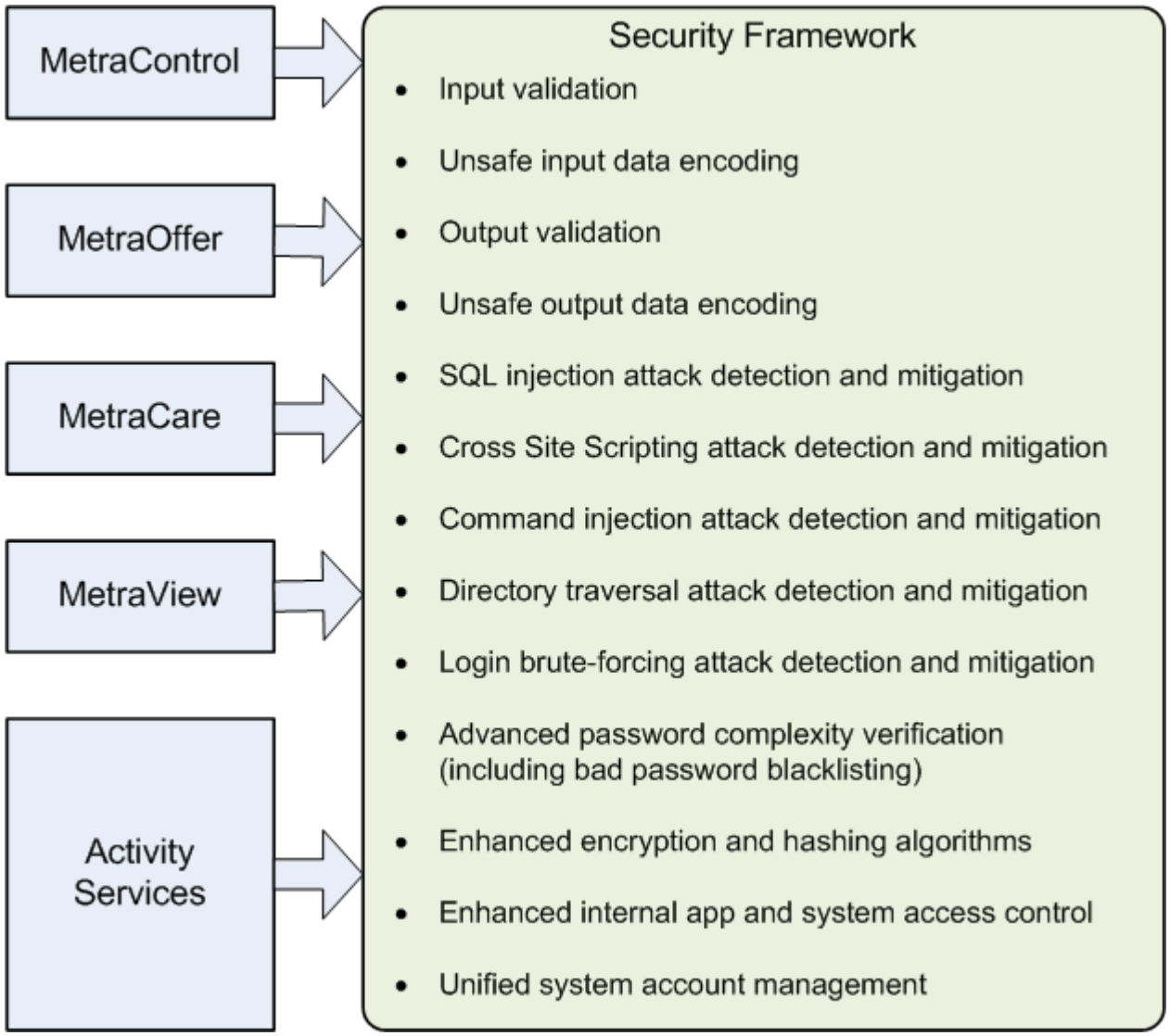
Security Framework Overview

Author: Kyle C. Quest

March 1st, 2010

metratech
Why Compromise?

HIGH LEVEL PRODUCT INTEGRATION OVERVIEW



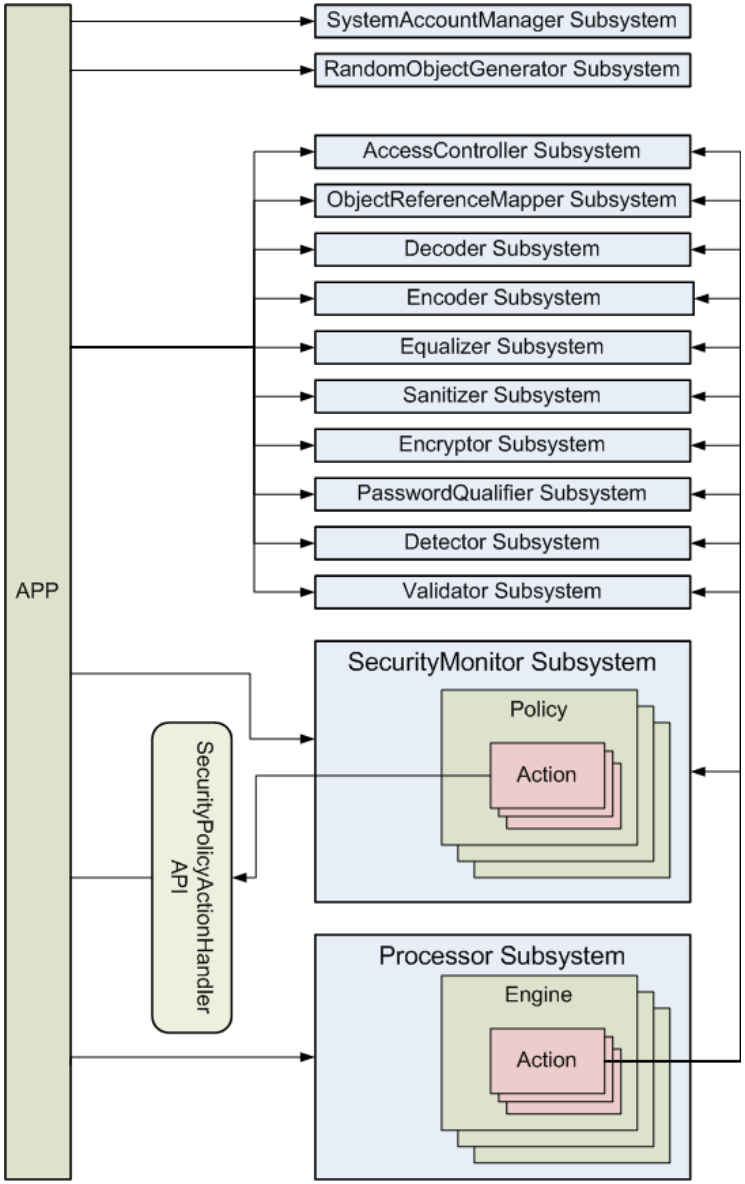
KEY LEGACY PRODUCT PROBLEMS

- Partial/insufficient/ad-hoc input data validation and encoding
- Insufficient output data validation and encoding
- Some internal application resources are exposed to external users allowing them to access or modify resources they don't own
- Insufficient access control
- Hard to change default accounts and passwords

KEY SECURITY FRAMEWORK BENEFITS (FOR PCI-DSS)

- Built-in web application firewall functionality satisfies PCI-DSS requirement 1 (use firewalls)
- Core functionality satisfies PCI-DSS requirement 6 (develop secure apps)
- Unified system account management simplifies PCI-DSS requirement 2 compliance (don’t use default passwords and account settings)

HIGH LEVEL ARCHITECTURE OVERVIEW



KEY API USE CONCEPTS

- Flexible API that provides access to low level security functions while providing high level processing sequence functionality
- Processor Subsystem provides the ability to chain together multiple processing actions including the ability to automatically report security events to the SecurityMonitor Subsystem
- SecurityMonitor Subsystem is configured with a set of policies that define how to respond to various security events.
- Application can register security policy action handlers with the SecurityMonitor subsystem to execute policy actions
- Policy actions include: blocking operation, blocking user, blocking address, logging, logging user out, sending security warning, etc

Detector Subsystem Use Example (SQL injection)

- **Initialize the Security Kernel during application startup:**
  - SecurityKernel.Initialize(<PathToSfPropertiesXmlFile>);
  - SecurityKernel.Start();
- **Call the Detector Subsystem:**
  - string testInputParam = "500' OR 1=1--";
  - testInputParam.DetectSql();
- **If the inspected data is not safe report a security event to the Security Monitor Subsystem:**
  - Catch(DetectorInputDataException x)
  - { x.Report(); }
- **Shutdown the Security Kernel:**
  - SecurityKernel.Stop();
  - SecurityKernel.Shutdown();

Encoder Subsystem Use Example

Call the Encoder Subsystem:

- string test = "<script>alert('hi')</script>";
- test.EncodeForUrl();
- Result => %3cscript%3ealert%28%27hi%27%29%3c%2fscript%3e
- test.EncodeForHtml();
- Result => <script>alert('hi')</script>
- test.EncodeForHtmlAttribute();
- Result => <script>alert('hi')</script>
- test.EncodeForCss();
- Result => \003cscript\003ealert\0028\0027hi\0027\0029\003c\002fscript\003e
- test.EncodeForJavaScript();
- Result => '\x3cscript\x3ealert\x28\x27hi\x27\x29\x3c\x2fscript\x3e'
- test.EncodeForXml();
- Result => <script>alert('hi')</script>
- test.EncodeForXmlAttribute();
- Result => <script>alert('hi')</script>