Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.24, NO.4, Aug. 2014 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2014.24.4.669

스마트 금융을 위한 비즈니스 로직과 분리된 보안프레임워크

서 동 현,† 이 상 진‡
고려대학교 정보보호대학원

A Business-Logic Separated Security Framework for Smart Banking

Dong-hyun Seo,† Sang-jin Lee‡
Korea Graduate School of Information Security

요 약

본 연구에서는 스마트금융 서비스를 위한 서버 측 보안 프레임워크를 제안한다. 국내 금융기관의 전자금융서비스를 제공하는 서버 측 프레임워크 구조를 살펴보면 대부분 서비스제공위주의 구조를 가지고 있다. 따라서, 보안관련 요구사항들은 비즈니스 로직들에 같이 포함되어 있는 경우가 대부분이기 때문에 보안 사고에 효과적으로 대응하기 어렵다. 본 논문에서는 전자금융서비스시 보안영역을 비즈니스영역과 분리하여 업무에 대한 의존도(Dependency) 없이 보안 정책을 실시간으로 적용할 수 있는 프레임워크를 제안한다. 이를 통하여 보안관련 위협에 대한 신속하고 효과적인 대응 기반을 제시한다. 또한 현재 서비스하고 있는 시스템구조에서도 시스템의 큰 변경없이 제안 프레임워크를 적용할 수 있는 방안을 제시한다.

ABSTRACT

This study introduces server-side security-oriented framework for smart financial service. Most of domestic financial institutions providing e-banking services have employed server-side framework which implement service-oriented architecture. Because such architecture accommodates business and security requirements at the same time, institutions are struggling to cope with the security incidents efficiently. The thesis suggests that separating security areas from business areas in the frameworks makes users to be able to apply security policies in real time without considering how these policies may affect business transactions. Security-oriented frameworks support rapid and effective countermeasures against security threats. Furthermore, plans to avoid significant changes on existing system when institutions implement these frameworks are discussed in the report.

Keywords: Smart Banking, Security Framework, Server Security Architecture

I. 서 론

최근 전자금융거래의 증가량이 가히 폭발적이라 할 만하다. 그중에서 모바일 뱅킹이 더 그러하다. 한국은행의 보도자료를 보면 2013년말 현재 인터넷뱅킹 서비스(모바일뱅킹 포함) 등록고객 수는 9,549만 명으로 전년 말 대비 10.5% 증가하였고 이중 스마트폰 기반 모바일뱅킹 등록고객 수는 전년 말보다 55.2% 증가한 3,719만 명이나 된다. 거래량으로 보면 증가추세는 더 크다. 2013년중 인터넷뱅킹 이용건수 및 금액(일평균 기준)은 전년대비 각각 18.7%, 1.3% 증가하였으며, 이 가운데 스마트폰뱅킹 이용건수와 금액이 각각 66.5%, 59% 증가하였다[1].

이러한 거래량의 증가추세와 함께 각종 보안사고의 위협 또한 증가하고 있는 것이 현실이다. KISA에서 발표하는 인터넷 침해사고 통계를 보면 해킹사고 접수

접수일(2014년 4월 21일), 게제확정일(2014년 7월 24일)
† 주저자, donghyun.seo@kbfg.com
‡ 교신저자, sangjin@korea.ac.kr(Corresponding author)

처리 현황은 매월 거의 1000건에 육박하고 국내 피싱 사이트 차단현황도 2012년에 금융기관관련 차단이 4,242건이었고 2013년은 5,940건으로 40% 가까이 증가한 것을 알 수 있다. 또한 악성코드를 유포하는 사이트도 2012년이 13,018건이었고 2013년에는 17,750건이 발견되었다[2].

이렇듯 보안위협의 증가는 실질적인 사고를 유발하고 각종 사고가 발생할 때마다 자체적인 대응 및 금감원, 금융위 등의 각종 권고, 규정, 지침 등이 계속 추가되거나 변경되고 이에 따라 개발 부서는 해당 정책을 구현하기 위해서 끊임없이 금융거래 프로그램을 수정하고 있는 것이 현실이다. Table 1.의 조사내역에서 보듯이 보안 관련한 변경이 자주 발생하고 있다.

안정성이 우선되어야 할 금융서비스가 비즈니스 요건과 더불어 보안요구사항으로 인하여 잦은 변경이 일어날 수 밖에 없고 개발 담당자들은 개발 및 테스트에 많은 부담이 될 수 밖에 없다. 이런 상황에 좀 더 적절하게 대응하기 위하여 서비스 개발을 위한 프레임워크가 필요하듯이 전자금융 서비스 보안을 위하여 특화된 프레임워크를 본 논문에서 제안한다.

이 프레임워크의 주된 목적은 전자금융서비스 영역과 보안영역에 대한 레이어(Layer)를 분리하여 금융서비스와 종속도(dependency)가 없는 보안 요구사항에 대하여는 보안 레이어에서 독립적으로 구현할 수 있는 아키텍처를 제공하는 것이다. 이렇게 함으로써 보안요구사항 적용시 서비스하고 있는 거래 프로그램의 수정 없이 개발할 수 있게 된다. 추가로 개발부서의 R&R(Role & Responsibility)도 보안개발파트와 서비스 개발파트로 분리하여 운영할 수 있게 되는데 점점 보안의 역할이 커지고 있기 때문에 점차 이런 역할분리의 요구 또한 증가되리라 생각된다.

Table 1. Security related application details of Bank A

Application Details	Impl Date
Strengthening self-identification for preventing e-banking incidents (Self)	13' 09
Applying services for preventing any e-banking fraud (Financial Supervisory Service (FSS))	13' 09
Discontinuing SMS authentication for preventing the e-banking fraud (Self)	13' 11
Strengthening the checking activities for any suspicious hacking transactions (Self))	13' 11
Strengthening the activities related to ARS certification (Self)	13' 11
Strengthening the registration process of E-banking PC (Self)	14' 01
IStrengthening the activities related to ARS certification (Self)	14' 01
Strengthening the services for preventing any E-banking fraud (FSS)	14' 02

여기에 전자금융서비스 구축시 기본적으로 요구되는 보안 준수사항에 대하여서는 기본 기능으로 제공하여 신규 구축시에는 빠른 구축이 가능하도록 지원하고 기존 서비스에 구축시는 커스터마이징을 쉽게 할 수 있는 구조를 제공하는 것을 목적으로 한다.

또한 본 논문에서는 보안프레임워크와 연계하여 모든 보안정책을 중앙에서 관리하는 Security Control Center 구축을 제안한다. 현재 서비스되고 있는 거래들에 대해서 보안정책들이 적용되어 있는 현황을 한눈에 볼 수 있는 대쉬보드(Dash Board)와 각종 정책들을 손쉽게 관리할 수 있는 기능을 제공해 보안담당자가 중앙에서 통합 관리할 수 있도록 하는 개념이 Security Control Center이다. 물론 이 부분은 사이트 별로 요구사항에 따라 별도로 구축되어야 하는 부분이 많은 영역이기 때문에 본 논문에서는 Security Control Center가 갖추어야 할 기본적인 기능과 공통 요구사항에 대해서만 다루도록 한다.

본 논문의 구성은 다음과 같다. 2장에서는 제안하는 보안 프레임워크에 대한 요구사항 도출, 아키텍처 설계에 대한 설명을 한다. 3장에서는 이 보안 프레임워크의 구현 및 적용 방안에 대하여 설명하고, 마지막으로 4장에서 요약과 함께 결론을 내린다.

II. 보안 프레임워크 요구사항

본 장에서는 스마트폰이나 인터넷을 통한 금융서비스를 제공할 때 필요한 보안 요구사항을 법과 제도적인 측면으로 분석해 보고, 현재 서비스 상의 보안관련 기능을 분석하여 보안 프레임워크에서 필요한 요건을 도출하고자 한다.

2.1 금융서비스 상의 보안 요구사항 분석

스마트 금융서비스를 제공함에 있어 보안 관련 요구사항이 나오는 부분은 크게 전자금융 관련 법・제도・규정 등을 준수하기 위한 것과 보안관련 주무부서의

별도 요구사항이다. 이중 전자금융 관련 법과 제도 쪽으로 보면 전자금융감독규정, 정보통신망 이용촉진 및 정보 보호 등에 관한 법률, ISO27001, OWASP등이 있는데 이 부분들은 각 종 보안 점검이나 모의해킹 등에서 근기로 삼고 있는 부분들로 꼭 준수해야 한다. 이중에서 개별 업무와 상관없이 보안 프레임워크에서 독립적으로 처리해 줄 수 있는 요구사항들을 추려 보면 Table 2.와 같다[4-8].

또한, 현재 제1금융권에서 제공하고 있는 보안관련 서비스 중 금융서비스 영역과 분리하여 보안프레임워크 영역에서 독립적으로 처리 가능한 보안 요구사항들에 대한 분석을 해보면 Table 3.과 같다. 주로 조회·이체 등의 대고객 서비스와 밀접한 관련 없이 처리 가능한 요건들이 이에 해당된다.

2.2 보안 프레임워크 요구사항 도출

스마트 금융에서의 보안요구사항들에 대한 조사를 토대로 보안프레임워크가 기본적으로 갖추어야 할 기능은 크게 보면 아래와 같이 4가지 정도로 도출될 수 있다.

첫째, 일반적으로 스마트금융서비스를 구축하기 위하여 필요한 보안요구사항 중 업무와 독립적으로 구현될 수 있는 기본 기능은 제공해야 한다. 그래야만 금융서비스를 구축할 때, 보안 부분을 빠르고 안정적으로 구축할 수 있다. 물론 각 모듈들은 상황에 맞게 커스터마이징(Customizing)이 가능한 구조를 가져야 한다. 기본 기능으로 제공될 수 있는 항목들을 살펴보면 거래 및 단말정보 수집, 거래 제한, 이중로그인 방지(세션탈취방지), XSS(Cross Site Scripting) 방어, SQL Injection 방어, 표준 암·복호화 모듈 등이다.

둘째, 보안 요구사항에 의해 만들어지거나 제공된 모듈에 대한 관리기능이 제공되어야 한다. 일반적으로 서비스 운영시 현재 작동하고 있는 보안모듈이나 추가로 만들어진 모듈에 대하여 작동 중지, 작동시킬 대상의 확대·축소, 신규모듈에 대한 즉각적인 적용등 다양한 상황이 발생할 수 있다. 따라서, 이와 같은 상황이 발생하면 보안담당자가 즉각적으로 대처할 수 있는 구조를 지원해 줄 필요가 있다.

Table 2. Possible implementation of the security provisions in the framework-level entry

Requirement	related laws/regulations
Must apply the encrypted communication when applied to e-banking transactions - E2E application : End-to-End encryption method to prevent the disclosure of sensitive self-authentication information such as Password, OTP that are entered in the client side(Keyboard Security Solution, Virtual Keyboard)	- ISO27001 A.10.9.2
When encrypting the critical information with using the national certified encryption algorithm, must use the encryption algorithm that are certified/accredited by the national institutions or equivalent - Secret key encryption : must use the secure algorithm SEED of more than 128 bit or equivalent - Public key : must use the RSA algorithm of more than 1024 bits or equivalent	- Article 11 Enforcement Regulations for electronic banking supervision regulations
One ID-One Session - Prevents from connecting to the online session with one ID at the same time and from different PCs or IPs - disconnects the previous session when the user moves the location from one to another or another user tries to login different id	- ISO27001 A.11.5.2
SQL Injection defence	- OWASP A1
Covering Session Management Vulnerability - Threats to hack the session tokens or attack the other implementation vulnerabilities by masquerading as a different user ID	- OWASP A2
XSS(Cross Site Scripting) defence	- OWASP A3

Table 3. Analysis of the security-related features that are implemented in the current financial sector

Category	Description	Separate(Y/N)
BlackList Management	Blocking transaction performed on the PC by managing the information of that PC that is used for hacking or has any suspicious activities	Y
Prevention of stealing/ reusing the security card number	Requesting ARS certification when user tries any transactions (Security card user only)	N
Restrictions to the registered user's PC/phone	Limiting services to deal only with user's registered PC or Phone	N
Prevention of the session hijacking	Comparing the information of the PC stored at login to the information of the PC that is requesting any transactions or inquiries	Y
Log Collection/Analysis	Collecting any transaction related information like terminal information for analysis on incident reports or any suspicious transactions	Y
Prevention of the dual login	Preventing the login success from different PCs and smart devices at the same time	Y
Communication encryption between the server and the client	Encrypting the communication between the server and the client (E2E)	Y
Extra authentication services such as ARS and SMS	Adding extra Authentication Services for identification of the customer to proceed any transactions	N
ARS, SMS Add Authentication Service	Add Authentication Service for identification of customer transactions	N
Keyboard Security / Security Keypad	Using the keyboard security and security keypad derived according to the customer's environment	N

셋째, 서비스 안정성의 측면이 고려되어야 한다. 본 프레임워크를 현재 서비스 중인 시스템에도 큰 리스크(Risk)없이 적용할 수 있어야 하며, 보안모듈이 신규로 적용되거나 변경되더라도 기본적인 서비스에 영향이 최소화 되도록 해야 하고 만일 서비스에 장애가 될 경우 즉시 해당 모듈이 정지될 수 있는 등의 안전망이 갖추어져야 한다.

넷째, 보안 통제 센터가 필요하다. 이 개념은 보안 프레임워크를 통제하는 센터로 현재 서비스하고 있는 모든 금융서비스 목록과 보안 모듈을 관리하고 현재 각 서비스 별 적용되어 있는 보안 모듈들을 한눈에 볼 수 있는 기능이 제공돼야 한다. 이것을 토대로 보안담당자는 현재 적용되어 있는 보안모듈들을 한눈에 볼 수 있고 서비스에 새로운 모듈을 추가하거나 삭제하는 것을 손쉽게 수행할 수 있다. 보통 어떤 보안 요건을 적용하기 위해서 보안요구사항을 내는 부서가 개발부서에게 개발을 의뢰하는 형태로 진행이 되기 때문에 전체 서비스 중에서 어떤 보안 정책이 어떻게 적용되어 있는지 현황을 관리하기 쉽지 않은게 현실이기 때문에 관리적인 측면에서 꼭 필요한 기능이다. 또한, 각 보안모듈에서 수집하는 각종 거래로그, 접속단말 로그 등이 모이는 곳으로 별도의 분석툴과 연계등을 통하여 다각적인 분석을 가능하도록 한다. 한 예로 FDS(Fraud Detection System)과 연동하여 거래 로그를 분석하고 또한 FDS를 통하여 거래를 실시간으로 통제할 필요가 있을 경우에 손쉽게 연동할 수 있게 된다. 물론 보안모듈을 생성・수정・배포 등의 관리기능은 기본적으로 제공해야 한다.

III. 보안 프레임워크 설계

3.1 프레임워크 기본 아키텍처

2장에서 도출된 요구사항을 기반으로 하여 프레임

워크의 기능을 크게 구분하여 보면 보안모듈을 처리하는 엔진 역할 부분, 표준 보안모듈 제공부분, 보안통제센터부분으로 나눠볼 수 있다. 또한 금융권 전반에서 사용할 수 있도록 현재 금융권의 인터넷뱅킹 및 스마트폰 뱅킹서비스 아키텍처인 J2EE(Java 2 Enterprise Edition) 기반으로 설계하여 이 기반위에서 손쉽게 적용할 수 있도록 하는 것을 목표로 하였다. 다음 고려사항으로 현재 서비스를 제공하고 있는 도메인에서도 이 보안프레임워크를 순차 적용할 수 있도록 보안프레임워크의 엔진은 J2EE 스펙중에서 Servlet Filter[3] 개념을 사용하여 설계를 진행하였다. 이를 토대로 전반적인 아키텍처를 그려 보면 Fig.1.과 같다.

현재 아키텍처에서는 스마트폰거래 혹은 PC등에서 HTTPS(Hypertext Transfer Protocol over Secure Socket Layer)를 사용하여 서버와 통신하는 것을 기준으로 하였다. 이 보안프레임워크는 전체 서비스 아키텍처에서 Presentation Layer에 적용되도록 구성하였고 Security Filter Engine 영역과 Security Control Center의 통신은 J2EE의 JMS(Java Message Service)[12]를 사용하여 서비스 중에 Control Center가 문제가 생겨도 JMS가 Queue 역할을 하여 서비스에는 문제가 생기지 않도록 하였다.

이 아키텍처에서 실질적으로 개발되거나 운영되는 보안모듈의 역할은 Security Filter이다. 기본기능으로 제공되거나 신규로 개발되는 최소단위 모듈로 Security Control Center를 통해서 관리되고 배포된다.

3.2 각 영역별 상세 설계

3.2.1 Security Filter Engine

이 부분은 Fig.1.에서 보듯이 다시 Filter Manager 영역과 Security Filter 영역으로 나뉘어진다. Filter Manager는 모든 Security Filter를 관리하는 Engine으로 다음과 같은 주요기능을 수행한다.

첫째, Filter 관리 기능이다. 이 기능은 Filter의 배포, 변경, 삭제, 기동, 정지 등의 작업을 Center의 요청을 받아 수행한다.

둘째, Filter의 작동 및 안정성 보장 기능이다. 이 부분은 Filter의 작동을 수행하고 혹시 Filter가 오류를 발생하더라도 기존의 서비스에 영향이 없도록 방어를 해주는 기능이다.

셋째, 통신 인터페이스를 제공한다. Filter가 Security Control Center와 통신할 수 있는 표준

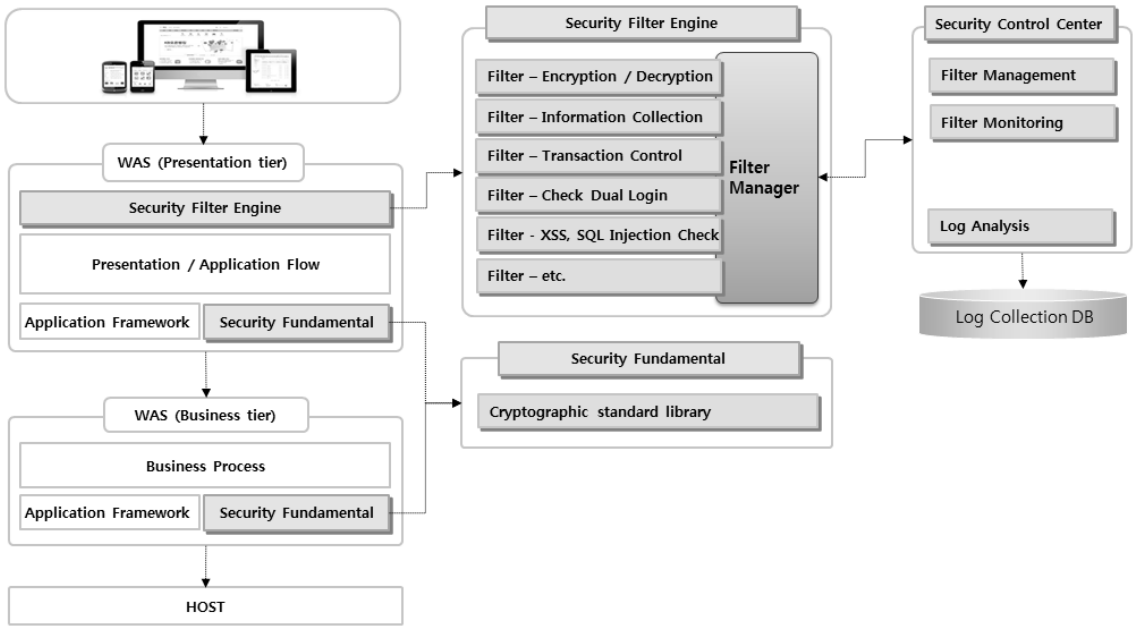

Fig. 1. Security Framework Architecture

인터페이스를 제공하여 Filter 개발자가 손쉽게 사용할 수 있도록 지원한다.

넷째, Filter에 대한 모니터링을 수행한다. 각 Filter에 대한 정상작동, 오류 등의 정보를 Control Center에 보고하여 중앙에서 모니터링을 할 수 있도록 한다.

다음으로 Security Filter는 보안 요건을 처리하는 최소단위로 Manager에서 정하는 표준 인터페이스로 개발되는 프로그램 가능한 모듈이다. 다시 말하면 여기서 Filter는 보안요구사항에서 논의된 관리되어야 할 하나하나의 보안정책을 의미한다. 조회나 이체등의 금융거래에서 전처리나 후처리에서 마치 Filter처럼 끼웠다가 뺄 수 있다는 의미와 함께 Servlet Filter SPEC을 사용하므로 이 같이 명명하였다.

이 각각의 Filter는 클라이언트로 부터 넘어오는 모든 파라메터 및 세션정보 등에 엑세스 할 수 있어 해당 업무 도메인에 특화된 보안정책을 수행할 수 있다. 따라서, 방화벽이나 IDS (Intrusion Detection System), IPS (intrusion prevention system) 등의 보안관련 장비가 수용할 수 없는 어플리케이션 계층(Application Layer)의 보안 정책을 담당한다. 기본적으로 각 Filter는 Application 레벨로 구현이 가능하기 때문에 구현가능한 모든 요구사항을 수용할 수 있는 구조이다.

위 개념을 가지고 Manager와 Filter개념을 Class Diagram으로 그려보면 Fig.2.와 같다.

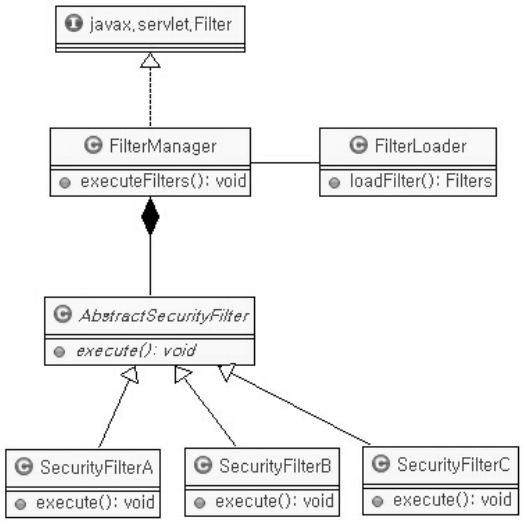

Fig. 2. Security Filter Engine Class Diagram

이 다이어그램에서 각각의 Filter는 Abstract SecurityFilter라는 추상클래스를 상속받아서 구현토록 하여 모든 Filter들이 일관된 인터페이스를 가지도록 강제하여 Manager에서 모든 필터를 실행시킬 수 있도록 한다. 또한 Control Center와 통신, 로깅처럼 Filter가 필요한 기본적인 공통 기능을 제공하여 각 Filter에서는 본연의 기능에 집중할 수 있도록 지원한다.

각각의 Filter는 하나의 클래스로 만들어지고 배포되는 구조를 가지고 Filter Manager에서는 해당 Filter Class를 실시간으로 로딩하여 처리할 수 있다. 이 부분은 FilterLoader가 처리하는데 Dynamic Class Loading[13]기술을 사용한다. 이것으로 Control Center에서 새로운 Filter를 Class로 만들어 배포하여도 운영 중에 실시간으로 적용하거나 변경하는 것이 가능해진다.

일반적인 금융거래 중 조회거래를 예로 들어 Security Filter Engine이 처리하는 개념을 Sequence Diagram으로 표현해 보면 Fig.3.과 같다. 여기서 Security Filter 는 고객의 접속단말정보를 수집하는 기능을 수행하는 Filter를 예로 들어 작성되었다.

이 그림에서와 같이 스마트폰 사용자가 조회거래를 요청하면 서버의 웹 서버가 수신하여 WAS(Web Application Server)로 거래를 전달하게 된다. WAS에서는 기동시 참조하는 web.xml 이라는 config파일을 참조하여 여기에 등록되어 있는 Servlet Filter를 실행시켜준다. WAS로 들어오는 모든 요청은 이 등록된 Filter를 거치도록 되어 있다. 여기서 보안프레임워크의 엔진인 Filter Manager가 작동하게 된다. Filter Manager는 Control Center에서 정의한 Filter와 Service Mapping

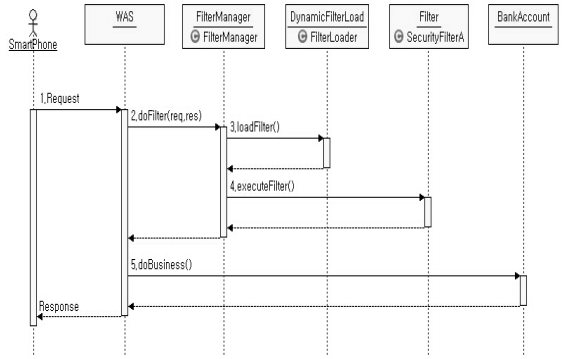

Fig. 3. Sequence Diagram

정보를 확인하여 현재 요청된 거래에 맞는 Filter List를 순차적으로 실행시키는데 FilterLoader에 요청하여 현재 버전의 Filter Class의 Instance를 받아서 수행시킨다. 그러면 해당 Filter에서 필요한 정보를 수집하고 수집된 정보를 Control Center에 전송하고 작업을 종료하면 Filter Manager가 추가적인 Filter가 있다면 반복적으로 이를 수행하고 없으면 원래 요청된 조회거래가 처리된다. 이렇게 됨으로써 기존의 조회거래에는 어떠한 수정이나 영향 없이 원하는 처리를 수행할 수 있게 된다.

3.2.2 Security Fundamental

이 영역은 금융권에서 개발시 필요로 하는 암복호화 관련 모듈을 프레임워크 차원에서 기본적으로 제공하는 개념으로 각 서비스 도메인별로 사용하는 보안모듈에 따라 추가되거나 변경될 수 있다. 일반적으로는 감독기관이 권장하는 비도의 표준 암호라이브러리를 제공하여 개별 업무에서 필요시 사용할 수 있도록 제공한다.

금융시스템에 암호 알고리즘을 적용하는 경우, 알고리즘의 종류나 키 길이 등은 해당 시스템의 안전성 수준을 만족할 수 있도록 선택하여야 한다. 이와 관련해서 미국, 유럽 등에서는 암호 알고리즘 및 키 길이에 대한 가이드라인을 제시하고 있고 국내에서도 이와 비슷하게 KISA에서 국산 암호 알고리즘을 포함해 보안강도에 따라 선택 가능한 암호 알고리즘의 종류와 키 길이, 유효기간을 제시하고 있다[4]. 이 기준에 충족되게 보안프레임워크에서는 개발자에게 사용할 수 있는 암호알고리즘을 제공하고 가이드를 제공한다. 기본적으로 제공하고 가이드하는 항목은 Table 4. 와 같다. 이렇게 함으로써 개발자가 안정성이나 보안강도에 대한 고려 없이 알고리즘을 선택하는 것을 방지하고 보안 관리자가 해당 모듈의 보안강도를 관리하도록 한다.

Table 4. Standard Security Algorithm

Category	Strength (bit)	Recommend	Duration
Symmetric key Algorithm	112~128	SEED HIGHT ARIA-128	112bit (2011~2030)
	192	ARIA-192	after 2030 (Max 30 Year)
	256	ARIA-256	
Hash	112	SHA-224	2011~2030
	128	SHA-256	after 2030 (Max 30 Year)
	192	SHA-384	
	256	SHA-512	

개발자입장에서는 개인정보 암,복호화시에 사용하는 모듈을 가이드하고 비밀번호등의 일방향 함수에 대한 가이드로서 제공되게 된다.

3.2.3. Security Control Center

이 Control Center는 위에서 언급한 요구사항대로 Filter에 대한 관리 및 Filter 작동에 대한 모니터링, 로그 수집 및 분석 처리를 수행하게 되는데 이 영역은 관리 툴 영역에 가깝기 때문에 구축하는 회사의 요구사항에 따라 커스터마이징(Customizing)하여 구현해야 하는 UI적인 요소가 많은 부분이기 때문에 본 논문에서는 Filter Manager와의 표준 인터페이스 방안과 기본적으로 갖추어야 할 기능적인 요소에 대해서 주로 다루도록 하겠다. 기본적으로 Filter와 Control Center간에는 Dependency를 최소화하고 또한 서로간의 상태에 따른 서비스에 영향을 주지 말아야 하기 때문에 메시지 방식으로 통신하는 것이 좋다. 메세징 시스템은 안정적이고 또한 확장 가능한 분산 어플리케이션을 제작하기 위해서 사용되는데 분리된 어플리케이션이 비동기적으로 신뢰성 있게 통신할 수 있게 해준다. 비즈니스에서 메세징 서비스를 이용하려면 MOM(Message Oriented Middleware)가 필요한데 JMS는 각 회사에서 만든 MOM 시스템의 공통적인 부분을 표준화함으로써 공통된 API를 사용할 수 있도록 한 것이다. 일반적으로 웹로직(Weblogic)과 같은 WAS(Web Application Server)는 MOM기능을 제공하고 있기 때문에 별다른 소프트웨어 설치 없이 사용할 수 있다.

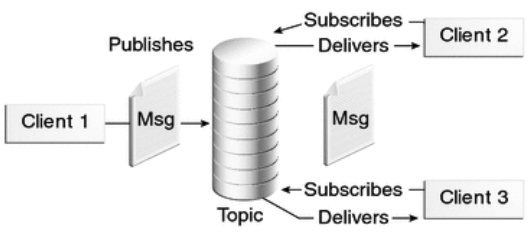

Fig. 4. Publishing/Subscribe Messaging

Security Control Center는 다수의 WAS Instance들과 통신하여야 하므로 JMS 중에서도 Publishing/Subscribe Messaging 방식으로 개별 인스턴스들에게 정책을 전달한다. Fig.4.처럼 개별 Instance들은 Control Center의 메시지를 구독하고 있다가 메시지가 발생하면 받아가는 형태이다. 이런 표준 메시지 전송 구조로 설계하면 Control Center와 Filter Engine 사이에 안정적으로 통신할 수 있게 된다.

다음으로 Security Control Center가 제공해야 할 기본적인 기능에 대해서 정리해 보면 다음과 같다.

첫째, Filter 관리 기능이다. 이것은 Filter의 등록, 수정, 삭제, 배포 등의 작업을 Center에서 관리할 수 있는 기능을 제공한다.

둘째, Filter 모니터링 기능이다. 이 부분은 Filter에서 전달되는 각종 이벤트 정보에 대한 Dashboard 형태의 현황판 및 조회 기능을 제공한다.

셋째, 거래서비스 관리영역이다. 현재 서비스 되고 있는 조회/이체등과 같은 거래들에 대한 관리기능을 제공하거나 기존에 정보가 있다면 연동하여 보여준다.

넷째, Filter 및 서비스 매핑 관리 이다. 이것은 매핑정보의 관리 기능 제공 및 매트릭스 형태의 통합 조회 기능을 제공한다.

마지막으로, 로그 분석 기능에 대한 제공이다. Filter에서 수집된 정보를 분석하는 기능 혹은 분석 툴과의 연동이 가능한 구조를 제공한다.

이중에서 서비스 매핑 관리에서는 서비스하고 있는 거래에 매핑되어 있는 Security Filter 뿐이 아니라 해당 거래가 준수해야 할 보안 정책도 같이 관리하여 보여준다면 현재 서비스되고 있는 모든 거래에 대해서 보안 정책을 한눈에 확인할 수 있으므로 그런 기능을 추가하는 것이 좋다. Table 5. 이 바로 그런 예시로 서비스와 보안정책에 대한 매트릭스 표이다.

실질적으로 작성한다면 인증수단항목도 더 많아질 것이며 고객정보입력 여부 등 좀 더 다양한 관리항목이 도출될 것이다. 금융거래를 서비스하는 조직은 바로 이런 정보를 현행화 해서 관리할 필요가 있으며 새로운 서비스를 준비할 경우는 Table 5과 같은 정보를 우선 작성해서 보안상 누락된 사항이 없는지 보안담당자들과 검토한 후, 개발을 진행하여야 한다.

서비스매핑 관리 이외에도 Security Filter에서 보내는 이벤트 정보에 대한 모니터링 및 분석 또한 중요하다. 가령 블랙리스트로 등록된 단말기의 거래 시도는 24시간 모니터링을 통하여 해당 거래를 시도한 ID가 무엇인지 확인하여 해당 고객정보에 대한 탈취, 해킹 여부를 확인하여 거래정지 처리등 여러 가지 조치를 취할 필요가 있기 때문이다. 이런 식으로 Security Control Center는 금융서비스를 제공하는 곳은 서비스 이용자의 금융자산을 보다 안전하게 지키기 위하여 꼭 갖추어야 할 시스템이며 조직이라 하겠다.

3.3 Security Filter 후보군 제안

다음은 금융서비스를 제공하는 곳에서 사용할 만한 Filter들을 정리한 것이다. 여기에 정리된 내역들은 일반적으로 금융서비스를 제공한다면 기본적으로 요구되는 보안 항목이 많으므로 신규 구축시에는 대부분 구현돼야 한다.

첫째, 클라이언트 보안모듈 연동 Filter 이다. 일반적으로 가상키패드, 키보드 보안모듈 등에서 암호화되어 올라오는 값들에 대한 복호화 처리를 하여 일반 거래 어플리케이션 영역에서는 신경 쓸 필요가 없도록 하여 준다.

Table 5. Service Security Check Matrix

Category		Certification					Security Filter					
		ID/ PWD	Cert	OTP	SecCard	ARS	E2E	Info collect	trans limit	dual login	SQL	XSS
Inquiry	accounts	O	O					O				
	transactions	O	O					O				
Transfer	instant		O	O	O	O	O	O	O	O	O	O
	auto		O	O	O	O	O	O	O	O	O	O
Cert	Issue / re-issue	O		O	O	O	O	O	O	O	O	O

둘째, 거래 및 단말정보 수집 Filter 이다. PC 및 모바일기기에서 올라오는 거래들에 대한 단말정보를 표준화 하여 수집하고 포렌식을 위한 거래로그를 기록한다. 기존 로그 수집 모듈 및 로그 분석 솔루션과의 연동도 가능하도록 한다.

셋째, 거래 제한 Filter 이다. 보안관리자가 단말정보/사용자식별정보 등을 Control Center에서 등록하여 거래를 통제할 수 있는 기능을 제공한다.

넷째, 이중로그인 방지(세션탈취방지) Filter이다. PC, 모바일기기등에서의 이중 로그인 시도가 발생할 경우 이를 차단하는 기능 제공 한다. 또한, 쿠키(Cookie)등의 탈취로 세션 하이제킹(hijacking) 시도에 대한 차단 기능으로 구현될 수 있다.

다섯째, XSS, SQL Injection 대응 Filter 이다. 사용자 단말에서 올라오는 값에 대하여 XSS, SQL Injection에 대한 대응 처리를 하여 개별 거래단에서 별도로 신경 쓸 필요가 없도록 한다(이 부분은 Filter에서 일괄 치환 등의 처리를 하고 개별 거래에서는 필요하다면 원래의 값을 확인할 수 있는 기능을 제공하여 개별 업무의 특수성을 고려한다).

마지막으로 실시간 위협 모니터링(이상징후 실시간 탐지) Filter 이다. 위에서 언급할 것과는 별도로 아래에 나열한 것과 같이 FDS 개념과 비슷한 것을 Filter를 통하여 개발할 수 있다. 이런 아이디어가 나오면 개별 비즈니스에 영향없이 적용하고 모니터링할 수 있다는 것이 제안 보안프레임워크의 최대 장점이라 하겠다.

IV. 보안 프레임워크 구현 및 적용

보안프레임워크에 대한 설계를 바탕으로 스마트폰의 고유한 단말정보를 수집하여 특정 단말에 대하여서는 거래를 거부하는 프로토타입 구현을 통해 실질적으로 작동하는 것을 검증해보고자 한다.

4.1 프로토타입 구현

본 프로토타입에서는 Security Filter Engine을 중점으로 구현해 보기로 하겠다. 프로토타입에서 구현될 Filter는 스마트폰에서 뱅킹앱으로 거래가 올라올 경우 스마트폰의 단말고유정보를 수집하여 현재 등록되 있는 거래제한단말인지 확인되면 거래중지 안내 페이지로 보내는 기능을 가지고 있다. 이 기능을 구현하려면 우선 사용자 단말쪽에서 모든 거래시 단말을 식

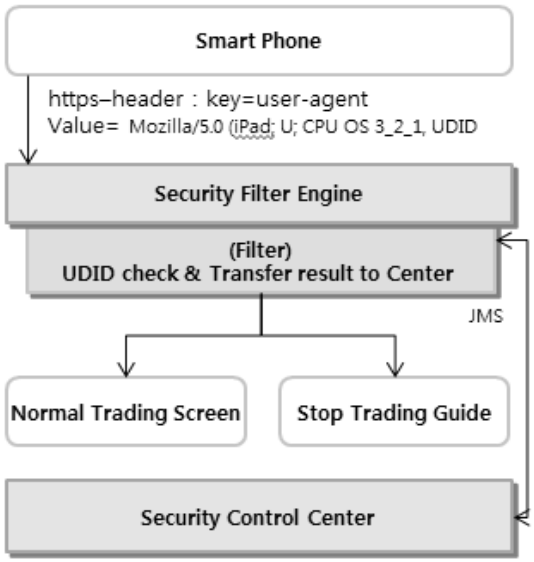

Fig. 5. Transaction Flow

별할 수 있는 고유정보를 보내주어야 하고 필터에서는 이 식별정보를 거래제한 단말인지 확인하고 맞다면 정해진 안내페이지로 보내는 기능이 구현되어야 한다. 우선 단말쪽에서 고유정보를 어떻게 모든 거래에서 서버로 전송할지에 대해서 살펴보겠다. 대부분의 사용자(약95%)가 ios/android 이므로 이 두 개의 OS를 기준으로 구현한다[9]. 우선 단말고유의 값을 정해야 하는데 ios/android 둘다 고유의 mac address를 가지고 있으므로 그 값으로 하였으나 ios7부터 mac address가 수집되지 않는 관계로 대신 apple에서 제공하는 identifierForVendor [10]로 사용하기로 한다.

다음으로는 이 값을 모든 거래시 서버로 전송하는 방법을 살펴보겠다. 앱 내의 개별거래에서 처리를 하거나 아니면 기존 앱의 통신프로토콜을 건드리면 수정사항이 많아지므로 가장 적은 수정으로 처리할 수 있는 방안으로 현재 사용하는 https 통신의 프로토콜 헤더의 User-Agent를 수정하는 것이 있다. 현재 User-Agent에 위에서 설명한 단말고유정보를 추가하여 서버에 전송하는 것이다. 이렇게 하면 앱 내에서도 통신을 처리하는 모듈만 수정하면 된다.

이제 서버 쪽 Filter의 구현에 대해 살펴 보면 위의 설계에서 보았듯이 Filter 실행시에 파라메터로 HttpRequest와 HttpResponse를 넘겨준다. 이 HttpRequest를 통하여 User-Agent값을 읽을 수 있게 된다. 이 값을 Parsing하여 거래제한단말 list

와 비교하여 일치하면 특정 안내 페이지로 보내면 되는데 Filter에서는 다음과 같은 형태로 거래흐름을 제어할 수 있다.

req.getRequestDispatcher(newURI).forward(req, res);

이런 방식으로 Filter가 동작 후 거래가 제한된 단말에 대해서는 Control Center로 해당 정보를 보내게 된다. 이것은 프레임워크에서 제공하는 api를 사용하여 JMS로 Control Center에 전달한다. 이렇게 함으로써 혹시 Control Center가 비정상이거나 응답지연이어도 뱅킹 거래에 영향을 주지 않고 서비스를 제공할 수 있다.

이렇게 구현된 내역을 간단히 그림으로 정리하여 보면 Fig.5. 와 같다.

여기서 구현해 본 바와 같이 거래단말제한이라는 요구사항을 구현하기 위해서 보안프레임워크 하에서는 스마트폰 쪽의 정보를 제공하는 부분과 단말정보 비교 로직 처리하는 Filter만 개발하여 Control Center에 등록하면 되는 비교적 간단한 작업이 된다.

4.2 구현 결과 분석

본 논문에서 제안한 보안 프레임워크 없이 위에서 구현한 것과 같은 기능을 하는 모듈을 구현하려고 한다면 개발자가 신경써야 하고 구현해야 할 항목이 늘어나게 되고 또한 새로운 요건이 나올때마다 비슷한 구현을 반복해야 할 것이다. 보안프레임워크를 사용하게 되면 보안요건에만 집중하면 되므로 보다 짧은 시간에 안정적으로 보안요구사항을 만족시킬 수 있을 것이다.

보안프레임워크 설계단계에서 살펴보았듯이 E2E 암호화, 스마트폰의 보안키패드 암복호화, 거래로그수집, 로그기록, 이중로그인 방지, XSS 대응 등을 제외하더라도 FDS와 같은 시스템을 구축할 때에도 기존의 서비스에 영향 없이 연동할 수 있는 최적의 솔루션이라 할 수 있다.

V. 결 론

현재 온라인상의 비대면 거래는 양적으로 엄청나게 성장하였다. 그에 따른 부작용으로 보안 관련 위협 또한 크게 증가하였다. 이에 온라인상에서 금융서비스를 제공하는 주체는 본연의 비즈니스만을 고려하여 서비스를 제공하는 것이 아니라 고객정보보호와 보안을 고려하여 서비스를 기획하고 제공하여야 한다. 이를 개발하고 지원하는 부서에서도 보안 관련 개발을 보다 효율적으로 하기 위한 방안으로 본 프레임워크를 제안하였다. 이 프레임워크는 비즈니스영역과 보안영역을 분리해보자는 개념을 가지고 현재 금융서비스에서 요구되고 있는 보안 관련 요구사항들을 분석하여 분리할 수 있는 요구사항들을 도출하였고 실질적인 프로토타입을 통해서 효율적으로 분리 구현이 가능하다는 사실도 입증하였다.

금융서비스는 안전성이 최우선이기 때문에 기존 서비스에 대한 영향이 최소화 되도록 설계단계부터 Filter 개념을 도입하여 기존에 서비스되고 있는 프로그램의 수정 없이 새로운 서비스에 대한 적용이 가능하도록 하며, 점진적인 적용 또한 문제없도록 하였다.

프로토타입에서는 거래하고 있는 스마트기기의 정보를 수집하고 통제하는 기능을 추가하는 것으로 수행하여 봤지만 FDS(Fraud Detection System) 같은 별도 솔루션과 연동하거나 아니면 직접 구축하는데 있어서도 효과적으로 응용할 수 있을 것이라 기대한다.

향후에는 온라인 금융서비스 개발을 지원하는 프레임워크를 현장에서 좀 더 검증하여 고도화 연구를 진행하고자 한다. 또한 개발부분의 보안 프레임워크를 좀 더 확장하여 금융서비스를 개발하는데 있어 보안 아키텍처 개념으로 연구범위를 넓혀 보고자 한다. 이렇게 함으로써 온라인 금융서비스를 구축하는데 있어 해당 업무 도메인에 특화된 보안 아키텍처 및 프레임워크를 제시하는 심화연구를 진행할 예정이다.

References

[1] The Bank of Korea, "Banking services usage statistics throughout the year 2013," The Bank of Korea, pp. 2-4, Feb. 2014.

[2] KISA, "Internet incident response statistics," KISA, pp. 133-139, Jan. 2014.

[3] Danny Coward and Yutaka Yoshida, "JavaTM Servlet Specification Version 2.4," Sun Microsystems, Inc, pp. 49-55, Nov. 2003.

[4] KISA, "Cryptographic algorithm and key length using guide," KISA, pp. 4-9, Jan. 2013.

[5] Financial Supervisory Service(FSS),

"Enforcement Regulations for electronic banking supervision regulations," Act 11,29, 2008.
〔6〕 Republic of Korea National Assembly, Legislation No. 09119, "Information and Communication Network Utilization and Information Protection Act," Act 28, 2008.
〔7〕 ISO, "Information security management A.10~11," ISO/IEC 27001, Oct. 2005.
〔8〕 OWASP Top 10 2013, https://www.owasp.org/index.php/Top_10_2013
〔9〕 Strategy Analytics, Android Captures Record 81 Percent Share of Global Smartphone Shipments in Q3 2013, http://blogs.strategyanalytics.com/WS S/post/2013/10/31/Android-Captures-Record-81-Percent-Share-of-Global-Smartphone-Shipments-in-Q3-2013.aspx
〔10〕 Apple Inc, MAC addresses in iOS 7, https://developer.apple.com/news/?id=8222013a
〔11〕 W3C, Header Field Definitions, http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html
〔12〕 Oracle Inc, Basic JMS API Concepts, http://docs.oracle.com/javaee/6/tutorial/doc/bncdx.html
〔13〕 wikipedia, Java Classloader, http://en.wikipedia.org/wiki/Java_Classloader

〈저자소개〉

서 동 현 (Dong Hyun Seo) 정회원
1998년 2월: 청주대학교 전자계산학과 졸업
2013년 3월~현재: 고려대학교 금융보안학과 석사과정
〈관심분야〉 정보보호, 아키텍처, 프레임워크

이 상 진 (Sangjin Lee) 종신회원
1989년 2월 ~ 1999년 2월: 한국전자통신연구원 선임 연구원
1999년 2월 ~ 2001년 8월: 고려대학교 자연과학대학 조교수
2001년 9월 ~ 현재: 고려대학교 정보보호대학원 교수
〈관심분야〉 대칭키 암호, 정보은닉이론, 컴퓨터 포렌식