홍 승 필 교수지도

석사학위청구논문

웹 환경 내 개인정보 공격에 대한 대응방안 연구

- 개인정보 위협 사례 연구 중심으로 -

2009

성신여자대학교 일반대학원

전산학과

김 혜 리

인 준 서

김혜리의 석사학위 논문으로 인준함.

심사위원 _____인

심사위원 _____인

심사위원 _____인

성신여자대학교 대학원

목 차

그 림 목 차

논 문 개 요

이전 웹 시스템 보안의 주 대상이었던 바이러스, 웜과는 달리 대부분 뚜렷한 목적을 가지고 사용자로부터 부당한 금전적 이익을 취하기 위하여 개인정보에 접근을 시도한다는 점에서 피싱(Phishing) 등의 금융 사기가 현재 보안의 핫 이슈로 대두되고 있다. 피싱이란 단어에는 비밀번호나 금융 정보와 같은 개인 정보를 낚으려는(fishing) 의도가 반영되어 있다. 피싱은 사회공학적 방법 및 기술적 은닉 기법을 이용하여 개인정보, 금융 계정 정보를 절도하는 수법이며 금융 사기가 전체의 78.4%에 달한다.

본 논문에서는 웹 환경 내의 개인정보 공격 방법과 사례(피싱사이트 등)를 분석하고, 사전에 예방할 수 있는 방안에 대해 제안하여 보았다. 제안된 웹 사이트 보안 수준 확인 시스템은 웹 페이지의 물리적, 내부적, 외부적 검사의 3가지 메커니즘으로 구성되어 있으며, 이를 통해 웹 사이트에 존재하는 위험 요소를 검사해 준다. 물리적 분석은 접속 중인 웹 페이지의 URL 주소 분석을 통해 이루어지며, 내부적 분석은 페이지 내 HTML 소스를 분석함으로써 다른 사이트로의 링크 주소와 Input Type Pattern 분석을 통한 위험 요소 검출이 이루어진다. 웹 사이트의 외부적 분석은 피싱 사이트에 사용되는 IP가 해외 사이트가 많다는 점을 바탕으로, 접속 중인 페이지의 서버가 국내에 존재하는지의 여부를 가지고 판단한다.

또한 접속 중인 웹 사이트의 신뢰도 검증을 위한 DB검색(WWL, WBL)을 통해 개인 정보 침해 사고를 최소화하는 방안을 제시하여 보았다.

Abstract

A Study on the prevention from personal information abuse in web environment - Focus on privacy threats case studying -

Kim, Hye Ri

Dept. of Computer Science
Graduate School
Sungshin Women's University

The financial trick like Phishing is currently to be a hot issue in the security areas. The reason to hot issue is phishing attempts to approach in private data with clear objects the unfairly financial profit from others, unlike virus or worm. Phishing is a form of online identity theft that employs both social engineering and technical subterfuge to steal consumers' personal identity data and financial account credentials. And 78.4% of phishing attacks targeted on financial services.

In this work, I analyze the ways and cases(Phishing etc.) of various personal information attacks in web environment, and suggested the prevention system in order to protect against from attacks. Web site Verification Management System consists of 3 mechanisms - physical, internal and external inspecting mechanism - and provides the threats alert existed inside web site with these mechanisms. Physical analysis comes out accessing web page's URL address, and internal analysis comes out inspecting linked address and input type patterns inside HTML source codes with the web page. And an external analysis of web site is decided by whether the server host exist domestic or overseas, because a lot of foreign web servers are used on phshing site many times.

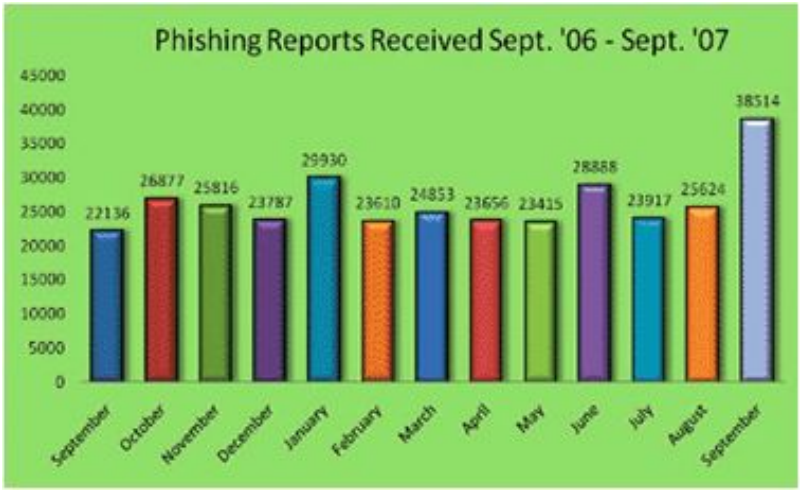
Finally I suggest that can be useful to reduce the Web site phishing threats with Website White List DB and Website Black List DB.

Ⅰ. 서론

1. 연구의 배경 및 목적

정보통신 기술의 발전으로 개인정보가 대량으로 수집되어 관리됨에 따라 불법적인 개인정보의 수집 및 유출은 특정 개인의 사회·경제적 활동에 치명적인 영향을 줄 뿐만 아니라 정보사회 자체에 대한 불신으로 이어질 수도 있다.[1] 개인정보란 “생존하는 개인에 관한 정보로서 성명·생년월일·주민등록번호 등에 의하여 당해 개인을 알아볼 수 있는 부호·문자·음성·음향 및 영상 등의 정보(당해 정보만으로는 특정 개인을 알아볼 수 없는 경우에도 다른 정보와 용이하게 결합하여 알아볼 수 있는 것을 포함한다.)”를 말한다.[2]

이전 웹 시스템 보안의 주 대상이었던 바이러스, 웜과는 달리 대부분 뚜렷한 목적을 가지고 사용자로부터 부당한 물질적·금전적 이익을 취하기 위하여 개인정보에 접근을 시도한다는 점에서 피싱(Phishing) 등의 금융 사기가 현재 보안의 핫 이슈로 대두되고 있다. 피싱은 개인정보(Private Data)와 낚시(Fishing)의 합성어로 해커들이 만든 용어이며 사회공학적 방법 및 기술적 은닉기법을 이용해서 민감한 개인정보, 금융계정 정보를 절도하는 신종 금융사기 수법이다. 유명기관을 사칭한 위장 이메일을 불특정 다수 이메일 사용자에게 전송하여 위장된 홈페이지로 유인하여 인터넷 상에서 민감한 개인의 금융정보를 획득하는 사회공학적 기법을 사용한다.[3]



[그림 1-1] 06년 9월 ~ 07년 9월 피싱 접수 건수
(출처 : APWG)

피싱은 개인의 중요한 정보를 부정하게 얻으려는 공격 시도이다. 1996년부터 쓰이기 시작한 피싱이란 단어에는 비밀번호나 신용카드 번호와 같은 개인 정보를 낚으려는(fishing) 의도가 반영되어 있다. 일반적으로 피싱 공격자는 전자 메일이나 메신저와 같은 전달 수단을 통해 신뢰할 수 있는 사람 또는 기업이 보낸 것처럼 가장된 메시지를 공격대상자에게 보낸다. 그리고 이를 통해 미리 공격자가 만들어 놓은 위장된 사이트로 들어온 공격대상자의 주민등록번호, 비밀번호, 그리고 금융 정보와 같은 기밀이 요구되는 개인 정보를 얻으려고 한다.

Gartner의 설문 조사 결과에 따르면 2007년 피싱 공격으로 인한 피해액이 30억 달러 이상이었다. 앞으로 인터넷 사용이 더욱 늘어남에 따라 피싱으로 인한 피해도 함께 증가할 것이다. 그리하여 본 논문에서는 다양한 피

싱 사이트의 공격 방법 및 유형을 분석하고, 많은 피싱 사이트들을 효율적으로 막을 수 있는 방안에 대해 제안하여 보았다. 이를 위해 제안된 시스템에서는 웹 시스템의 물리적, 내부적, 외부적인 요인을 분석하여 잠재되어 있는 공격의 위험에 대한 정보를 사용자에게 제공하여 피싱 공격으로부터 사전에 예방할 수 있도록 한다.[4]

본 논문의 구성은 다음과 같다. 1장에서는 논문의 개요에 대해 간략히 소개하였고, 2장 에서는 웹 시스템 환경 내 위험 요소를 알아보았으며, 3장에서는 2장 내용을 바탕으로 문제점을 도출하고 해결 방안을 제시하여 보았다. 4장에서는 앞 장의 분석 결과와 문제점을 바탕으로 웹 시스템 내 위험성 분석 및 관련 메커니즘을 제시하였고, 5장에서는 시스템의 상세 설계와 구현 결과를 정리하였으며 마지막으로 6장에서는 결론을 기술하였다.

Ⅱ. 웹 시스템 환경 내 위험 요소

1. 웹 시스템 취약점

1.1 개요

인터넷을 통한 금융거래, 쇼핑 등이 대중화됨에 따라 관련 신종 사기 수법들이 속속 등장하고 있다. 이러한 사건들은 전문적인 기술을 통하여 이루어지는 경우도 있지만, 사회공학적인 기법과 같이 다른 사람 또는 조직을 사칭하여 사용자로 하여금 특별한 의심 없이 어떠한 행위를 유도하여 피해를 준다.

피싱 공격은 보통 기술적인 기법과 사회공학적인 속임수를 혼합하여 사용한다. 피셔(Phisher)는 고객이 정상적으로 이용하는 사이트에서 보낸 메일로 위장(Spoofing)하여 메일을 받는 사용자가 편지내용 중에 링크를 클릭해 위장된 사이트로 연결되고, 사용자가 자신의 금융정보를 입력하면 해커는 자신의 이메일을 통해 전송 받거나 피셔의 신원을 감추기 위해 해킹한 특정 서버에 저장해 놓았다가 이를 다시 유출해가는 형태를 띤다. 이번 장에서는 웹 시스템의 취약점과 대표적인 개인정보 유출 공격인 피싱에 대해 자세히 알아보고자 한다.

1.1.1 사회공학적 공격의 정의

사회공학적 공격 기법이란, 고도의 기술이 접목된 해킹기술과는 전혀 무관한 것으로, 기술적인 방법을 이용하는 것이 아니라, 인간의 심리적인 면을 이용하여 개인 정보 또는 신용 정보와 같은 중요한 정보를 획득하거나, 타인 스스로가 악의적인 결과를 발생하는 행위를 하도록 유도하는 것을 말한다.[5] 대부분의 인터넷 사용자는 웹 브라우저 상의 메시지의 지시를 충실히 이행한다. 그러므로 악의적인 시스템 공격자는 웹 브라우저의 자바나 자바 스크립트를 사용하여 다음과 같은 메시지를 사용자에게 보여줌으로써 원하는 정보를 얻거나 시스템 공격을 할 수 있다.[6]

o 고객의 계좌에 문제가 있으니, 신용카드 번호와 암호를 입력해 주십시오.
o 당신이 사용하고 계신 브라우저는 오래된 버전의 것입니다. 다음 링크를 클릭하여 새로운 버전의 브라우저를 다운받으신 후 setup.exe를 실행시켜 주십시오(이때 사용자는 악의적으로 제작된 브라우저를 다운로드 받을 수 있다).

사회공학적인 공격 방법은 공격을 당하는 피해자가, 이와 같은 공격 형태에 대한 지식이 없는 경우, 쉽게 당할 수 있으며, 실제로 아직도 많이 활용되고 있는 공격 방법이다. 그러므로 사회공학적 공격에 대처하기 위해서는, 사회공학적 공격의 위험성을 주기적으로 사용자에게 인지시키며, 정보보호에 대한 의식을 향상할 수 있는 별도의 교육의 실시가 필수적이다.

1.1.2 개인정보 유출 공격의 정의

개인정보 유출 공격이란 다양한 방법을 통해 개인의 중요 정보를 획득하고, 획득한 타인의 개인정보를 악의적으로 이용하는 행위를 말한다. 이와 같은 개인 정보 유출은 고도의 해킹 기술을 이용하여 이루어지는 경우도 있으나, 사회 공학적 공격 방법을 이용하여 시도되는 경우가 대부분이다. 최근 가장 큰 문제가 되고 있는 공격으로는 피싱, 파밍 그리고 악성코드 등을 이용한 공격이 있다.[5]

1.2 공격 유형

1.2.1 사용자 측면

1) 이메일

Email을 이용한 피싱 공격은 가장 일반적이다. 피셔는 스패머들(Spammers)이 사용하는 기술과 도구를 이용하여 메일서버통신 프로토콜(SMTP) 내의 'Mail From:''RCPT To:' 헤더를 조작하여 발송자를 위장한다.

대부분의 사용자들은 Email로 자신의 민감한 정보(암호 또는 PIN 번호)를 전송하는 것에 매우 신중을 기하고 있지만 여전히 Email에 의한 피싱 사건이 빈번히 발생하고 있다. 다음은 Email에 의한 피싱 기술들이다.[7]

o 기업 로고 도용 등으로 공식적인 Email로 위장

o 쉽게 인식하지 못하도록 약간의 URL을 변경한 법인 이메일

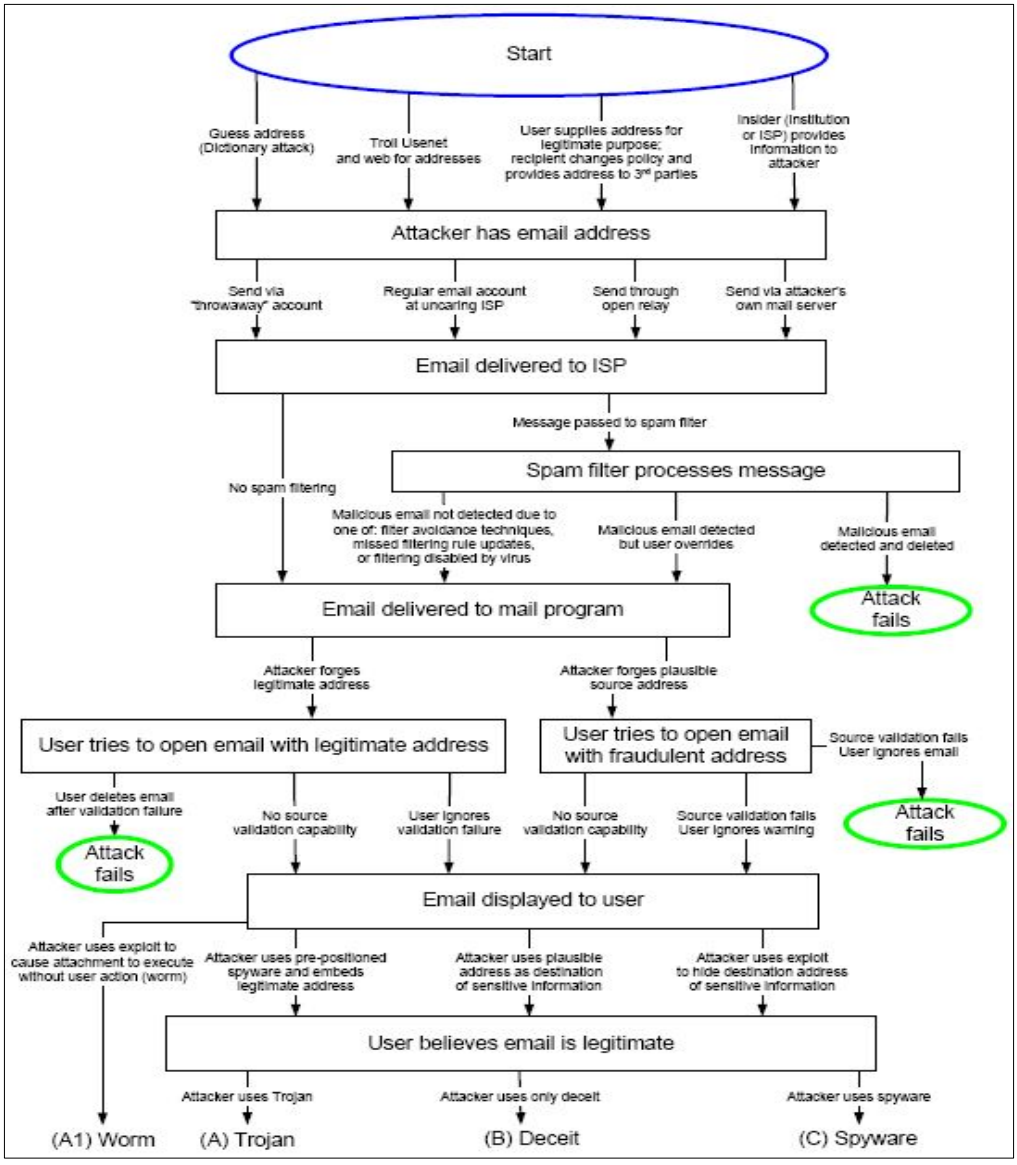
o URL 정보를 혼란스럽게 사용한 HTML 기반 이메일(비슷한 URL 스펠링 사용)

o 웜·바이러스를 첨부한 Email

o Spam 탐지를 우회하는 기법 이용

o 개인 신상을 담은 Email

o 인기 있는 게시판이나 메일링 리스트에서 보낸 메일로 위장



[그림 2-1] 전자메일을 통한 피싱 공격 흐름도

2) 메신저

IRC와 인터넷 메신저의 이용은 최근 새롭게 등장한 피싱 전달기법 이다. 또한 메신저를 이용한 커뮤니케이션 활용이 보편화되고, 메신저의 기능성이 향상되면서 메신저를 이용한 지능적인 피싱 공격이 증가하고 있으며 메신저가 그래픽, 멀티미디어 등 동적인 콘텐츠를 메신저 참여자에게 전송하는 기능이 추가되면서 웹기반의 피싱 공격 기법을 용이하게 이용하고 있다.

Bots 종류의 바이러스와 같은 형태로 전파되며, 피셔는 피싱 공격을 성공하기 위하여 사용자를 속이기 위한 다양한 트릭과 기술을 이용한다. 아래는 주로 사용하는 기술들이다.[8]

o 위장(Man-in-the-Middle) 공격
o URL 혼동 공격
o XSS(Cross Site Scripting) 공격
o 사전 세션 설정 공격
o 고객 자료 관찰 공격
o 클라이언트 취약점 이용 공격

1.2.2 시스템 및 서비스 측면

1) 웹사이트

Email과 더불어 가장 널리 이용되는 피싱 공격은 취약한 웹사이트 콘텐츠를 이용하는 것이다. 변조된 웹사이트를 이용한 피싱 방법이 점차 증가하면서 피싱 공격자가 직접 운영하는 웹사이트에 거짓 웹사이트 콘텐츠를 포함시키거나, 다른 사람이 운영하는 사이트를 활용한다.

웹을 기반으로 하는 피싱 메시지 전달방식은 아래의 기술을 주로 사용한다.[9]

o 인기 있는 웹사이트 혹은 게시판에 위장된 HTML 링크 삽입

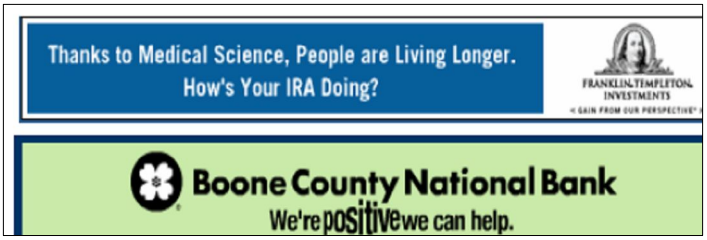
o 특정 웹사이트에 배너 이미지 등에 피셔 웹사이트 링크

o 숨겨진 아이템 또는 제로사이즈의 이미지와 같은 웹 버그 이용

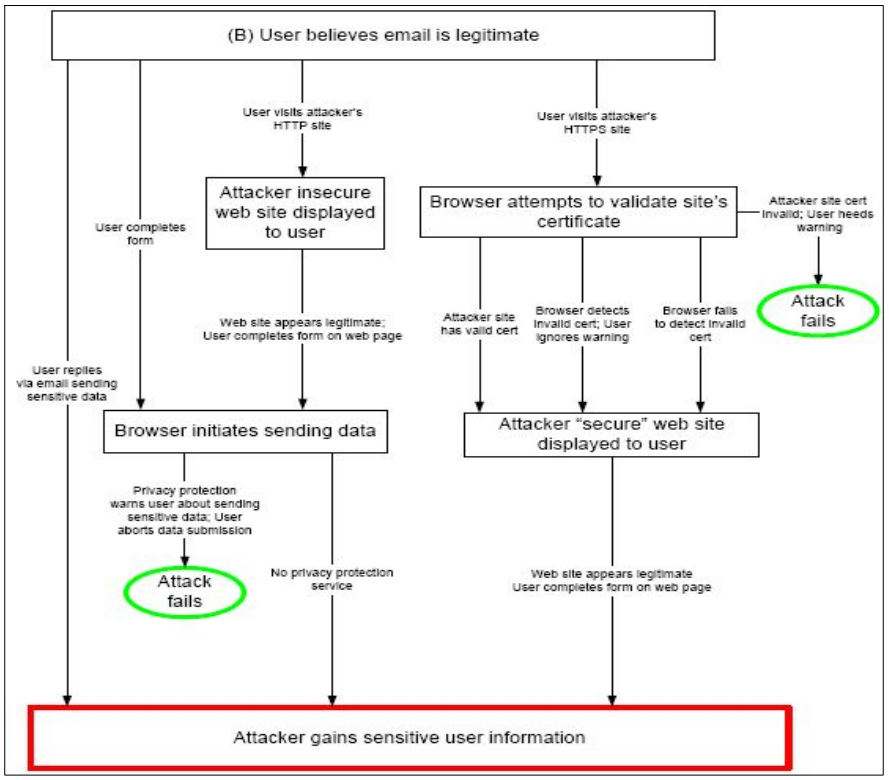
o 피셔 메시지의 주소를 위장하기 위해 팝업창 또는 프레임 사이즈가 제로인 윈도우 이용

o 웹브라우저의 취약점을 이용하여 악의적인 프로그램(Key-loggers, Screen-grabbers, Back-door, Trojan 등)을 사용자 PC에 설치

o 사이트의 관리자 권한에 의해 제공되는 컴포넌트 또는 데이터를 변조하여 이용자와 웹사이트의 신뢰관계를 이용



[그림 2-2] 웹 기반 피싱 예



[그림 2-3] 변조된 웹 페이지에 의한 피싱 공격

2) 트로이목마(Trojan Horse) 호스트

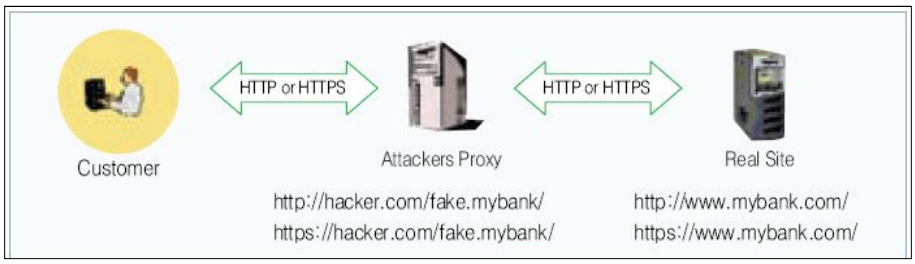
트로이목마에 감염된 PC를 통한 피싱으로 Key logger Trojan, Netbus와 같은 프로그램을 이용해서 공격자는 개인 PC 이용자가 입력하는 금융정보나 개인정보를 수집하거나, 공인인증서 파일을 유출하는 방식이다.

1.2.3 기술적 측면

1) 위장 공격 (Man-in-the-middle)

위장공격은 인터넷 이용자의 정보 및 자원에 대한 통제권을 획득하기 위해 공격자가 이용자 사이트와 대상 사이트의 중간에 자신의 사이트를 배치하여 커뮤니케이션의 조정자 역할을 수행하면서 모든 거래정보를 수집 및 관찰 하는 것으로 공격의 난이도가 높은 만큼 가장 성공률이 높은 공격방법이다. 피셔는 일반 고객과 실제 사이트의 중간에서 상호간의 통신을 중개하는 역할을 수행하면서 고객의 민감한 정보를 수집한다. 아래 (그림 2-4)는 위장 공격의 개념도이다.[10]

피셔는 고객과 사이트의 중간에서 ARP, IP, BGP 등과 같이 OSI 7 Layer 중 Layer 2, 3 프로토콜 취약점을 이용한다.



[그림 2-4] Man-in-the-middle-attack 구조

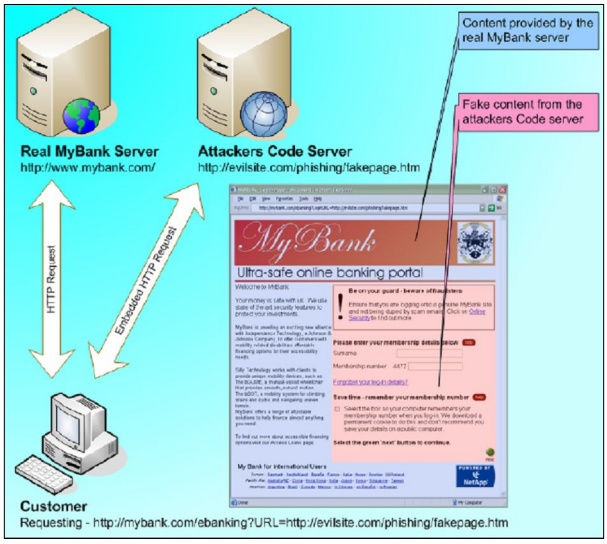
위장 공격을 성공시키기 위해서는 고객이 직접 공격자의 프락시 서버에 접속(고객은 실제 서버에 접속한 것으로 착각)하도록 하는 기술을 필요로 하며, 이에 대한 기술은 아래와 같다.

o Transparent Proxies
o DNS Cache 오염
o URL 혼동
o 브라우저의 프락시 서버 설정 변경(Browser Proxy Configuration)

2) URL 위장

대부분의 피싱 공격은 피해자들이 자신이 속았다는 것을 인식하지 못하게 하면서 피셔가 장악한 서버로 피해자의 정보를 보내게 하는 것이다. 피싱 공격자가 위장 도메인 명, 친숙한 Login URL, 제3자의 단축 URL 호스트 명 위장, URL 위장 등을 통해 메시지 수신자를 자신의 사이트로 유인하는

것으로 실제 사이트와 비슷한 스펠링 사용, DNS 오염 등과 같은 기법이 URL 혼동 기법이다.[8]



[그림 2-5] Cross-site scripting 공격 예

1.2.4 기타

피싱의 공격 기법 중 기술적인 측면에 아래 [표 2-1]과 같은 방법들도 있다.

[표 2-1] 기술적인 측면으로 본 피싱 공격 법

사전 세션 설정 (Preset Session) 공격	실제 사이트 고객에게 특정 세션 아이디가 삽입된 URL이 포함된 이메일을 발송하여 불특정 고객이 이 링크를 클릭하였을 경우 피싱 사이트에 로그인이 된다.

은닉(Hidden) 공격	공격자가 은닉 프레임, 웹페이지 콘텐츠 중복, 그래픽 교환을 통해 실제의 웹페이지 화면을 은닉 하고 위장된 웹페이지 화면을 인터넷 이용자에게 표시한다.
고객 자료 관찰 (Observing User Data) 공격	FTP, HTTP, SMTP 등과 같은 프로토콜(Layer 4 Protocol)의 취약점을 이용하여 Key-logger, Screengrabber와 같은 툴을 사용자 PC에 설치하여 민감한 정보를 수집할 수 있다.
클라이언트 취약점 이용 공격	웹 브라우저는 다양한 기능을 수행할수록 취약점 또한 증가한다. MS IE URL Mishandling, MS IE에 Media Player 기능 삽입으로 인한 취약점 등을 이용한 피싱 사건도 등장하고 있다.

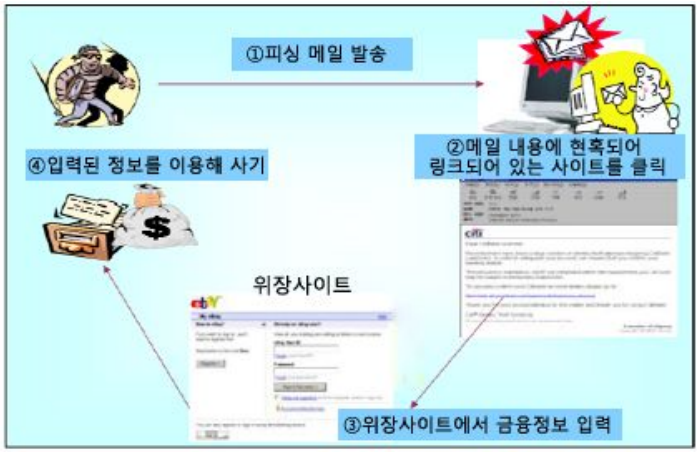
2. 개인정보 침해 사례 연구

2.1 피싱, 파밍

2.1.1 피싱

피싱(Phishing)은 개인정보(Private Data)와 낚시(Fishing)의 합성어로 해커들이 만든 용어이며 사회공학적 방법 및 기술적 은닉기법을 이용해서 민감한 개인정보, 금융계정 정보를 절도하는 신종 금융사기 수법이다. 피싱은 유명기관을 사칭한 위장 이메일을 불특정 다수 이메일 사용자에게 전송

하여 위장된 홈페이지로 유인하여 인터넷 상에서 신용카드 번호, 사용자 ID, PW 등 민감한 개인의 금융정보를 획득하는 사회 공학적 기법을 사용한다. 피싱은 금전적 피해를 유발하고, 공격 유형이 지속적으로 변화하며 피싱에 대한 대응이 매우 어려운 것이 특징이다. 피싱 피해는 계속 증가하고 있으며, 그 규모 또한 크게 증가하고 있는 상황이다. 피싱 공격 방법으로는 Man-in-the-Middle 공격, Cross-site-Scripting(CSS), URL 위장, 데이터 감시, 은닉방법 등이 있다.[11]

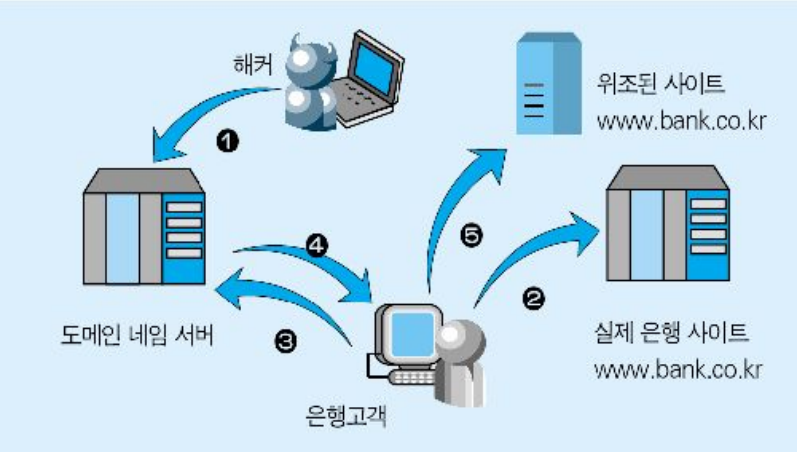


[그림 2-6] 피싱 발생 흐름도 예

2.1.2 파밍

파밍은 다양한 방법을 통해 정당한 사용자가 특정 도메인명에 대한 IP 주소 확인 요구 시, 해커가 장악하고 있는 악의적인 웹 사이트로 접속되도록 유도하는 공격 기법을 의미한다. 이메일을 통해 사용자 접속을 유도해 정보

를 빼내는 피싱과는 달리, 해당 사이트가 공식적으로 운영하고 있는 도메인 자체를 중간에서 탈취하는 파밍은 도메인 주소나 URL 주소로 진위여부를 파악하기 어려워 사용자들이 쉽게 알아차릴 수 없다는 점 때문에 더욱 강력한 위협 요인으로 작용하고 있다. 파밍은 순수 기술적인 공격 방법을 이용하여, 사용자가 해당 웹사이트가 정당한 것인지를 또는 악의적인 것인지를 파악할 수 없도록 하는 기법을 이용하는 것이다. 파밍은 주로 웹 트래픽의 전달방향 변경(redirection)을 통해 시도된다.[11]



[그림 2-7] 파밍 발생 흐름도 예

2.2 사례 연구

2.2.1 국내은행 해외지점 위장 피싱 사기 사이트

다음의 사례는 피싱 방법 중 가장 흔한 사례인 실제 사이트와 유사한

URL을 개설하여 가짜 사이트로 접근하게 하는 사이트이며, 자세한 내용은 다음의 신문 기사와 같다.

국내은행 해외지점 위장 피싱 사기사이트 기승

산은 런던지점 한달새 6개

피싱 사이트들은 모두 국내 은행의 해외지점을 가장한 피싱 사이트가 등장했다. 25일 관계업계에 따르면 최근 한달간 산업은행 런던지점을 사칭한 피싱 사이트 6개가 무더기로 발견됐다. 지난해 일부 시중은행의 국내 사이트를 모방한 가짜 사이트가 발견된 적은 있지만 해외지점 사이트를 모방한 피싱 사이트가 등장한 것은 산업은행의 영문 이니셜 'KDB'와 영국을 뜻하는 'UK'를 혼합해 런던 지점을 사칭했다.

(중략)

산은 관계자는 "가짜 사이트를 만들어 개인정보를 알아낸 뒤 범죄 등에 악용

할 목적으로 피싱 사이트를 개설한 것으로 보인다"며 "현재 금융결제원의 공동보안관제시스템(ISAC)과 국가사이버안전센터, 사이버수사대에 신고했으며 영국에서 수사를 진행하고 있다"고 말했다.

[디지털타임스 2006-08-28]

2.2.2 국내· 외 현황

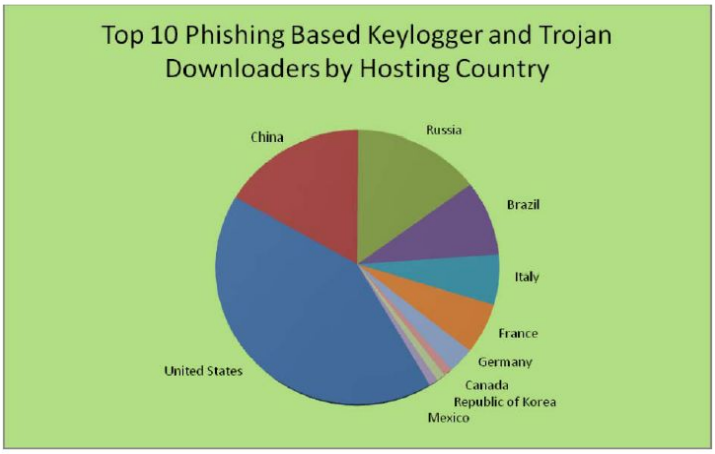
피싱 통계의 추세를 분석해보면 2007년 6월 발생한 피싱의 98.8%는 HTTP 80번 포트를 이용하고 있으며 오직 1%만이 80번 포트를 사용하지 않는 사이트였다. 이 기간 동안 28,888건의 새로운 피싱 보고가 리포트 되었고 31,709개의 피싱 사이트가 새롭게 생성되었다. 피싱 사이트는 평균 3.8일 정도 활동을 하였고 가장 긴 피싱 사이트인 경우 30일까지 활동한 사이트도 있었다. 2007년 1월 29,930건의 피싱 보고 이후 24,000여건 정도로 감소/정체되는 추세를 보이다가 6월 약 29,000건으로 상승하여 전달 대비 5,000건 이상 증가하였다.

피싱 사이트에 도용된 브랜드 중에는 금융기관이 95.2%로 가장 많았으며, AOL 등 ISP가 0.7%, eBay 등 소매상이 1.4%, 정부기관이 2.7%를 차지하고 있다. 특히 금융기관을 목표로 하는 피싱의 증가가 두드러지는데 금전적 이득을 취하기 위한 목적으로 활용되기 때문이다.[12]

피싱 웹사이트 호스팅 비율은 미국 31.95%로 가장 많았으며 중국, 일본 순서였다.

[그림 2-8] 10대 피싱 사이트 호스팅 국가 분포 [2007.07 APWG]

우리나라의 경우 피싱 사이트 호스팅 국가 순위에는 없으나 키로거나 트로이목마를 다운로드 받을 수 있는 피싱 사이트로는 10위권 내에 들고 있다.



[그림 2-9] 피싱 기반 키로거와 트로이목마 사이트 국가별 분포 [2007.07 APWG]

국내의 경우 KrCERT/CC의 통계에 따르면, 국내 피싱 경유지로 사용된 피해건수는 2006년 1,266건이었고 2007년 7월 현재 752건이 접수되었다. 월간 80여건 안팎으로 꾸준히 발생하고 있으며 2월 178건을 제외하고 2006년에 비해 월간 피해 접수건수는 감소하였다.[13]

국외도 마찬가지로 피싱 사이트에 도용된 브랜드 중 금융기관이 가장 많은 비중을 차지하였으나 전체 비율은 65.4%로 국내의 경우와는 큰 차이를 보인다. 특히 e-bay나 paypal과 같은 전자상거래의 비율이 33.3%를 차지하여 두드러진 차이점을 보인다.

Anti-Phishing Working Group은 매달 피싱 공격 증가율이 50%에 이르고 있으며, 피싱 공격의 지능화로 피싱 메일의 약 5%가 공격에 성공할 것이라고 보고한다. 그 이유는 아직까지 피싱에 대한 이해 수준이 미흡하고 피싱 방법이 갈수록 지능화 되어 가고 있기 때문이다. 이러한 문제에 대해 빠르게 대처하지 않으면 피싱의 도용사이트가 금융기관과 같은 금전적 이득을 목적으로 하는 경우가 많기 때문에 많은 피해가 발생할 것이고, 지금도 발생하고 있다.[14]

2.3 안티 피싱 솔루션 분석

안티 피싱 솔루션의 접근 방법은 피싱 사이트를 블랙리스트로 관리· 차단하는 방법과 반대로 신뢰할 수 있는 사이트 리스트를 기반으로 하는 화이트 리스트 방법이 있다. 블랙리스트 방법은 다양한 기업(AOL, Verisign)등에서 실시하고 있는데, 이미 공격의 피해가 발생해야 처리가 가능하다는 점과, 실

제 피싱 사이트의 수명이 몇 시간에 불과한 (Zero-Hour) 특성으로 볼 때, 현실적인 대응 안이 되지 못한다.

화이트리스트는 이와는 반대로 신뢰할 수 있는 사이트 리스트를 유지 하는 기법이다. 화이트리스트를 기반으로 웹사이트의 보안수준을 확인하고 있는 연구는 극도로 미약한 실정이며 아래는 현재 웹사이트의 보안수준을 확인하기 위해 국내에서 제공되고 있거나 개발되고 있는 안티 피싱 솔루션을 표로 정리한 것이다.[15]~[19]

[표 2-2] 국내·외 Anti Phishing Solution

	서비스명	DB List	기능
국내	노피싱 (NO PHISHING)	Black List	DNS 변경 방지, 파밍 방지
	노턴 (Norton Confidential)	Black List	패스워드 암호화, 키로거
	엔프로텍트 피싱헌터 (nProtect)	Black List	보호정보 패턴 정의
	시큐플랫 브이아이피 (SecuPlat VIP)		실시간 IP 모니터링
	클라이언트 피싱 프로 (Client PhishingPro)	Black List, White List	IP 식별, 키워드 필터링
국외	SpoofStick		Website의 SubDomaion의 Real DomainName 제공
	Netcraft Toolbar		도메인 등록 날짜, 호스팅 국가, 유명도 제공
	Trustbar		Secure web Connection (SSL), 인증권한

Ⅲ. 문제점 도출 및 해결 방안

현재 이용되고 있는 안티 피싱 솔루션들은 피싱 사고 이후 보고된 피싱 사이트를 확인하고 이를 등록하여 관리하는 블랙리스트기반 안티 피싱 솔루션과, 피싱 사이트가 아닌 안전한 사이트임을 등록하고 확인해 주는 화이트리스트 기반 안티 피싱 솔루션이 주를 이루고 있다. 대부분 안티 피싱 관련 솔루션의 경우, 블랙리스트를 이용하여 웹사이트 보안수준에 대한 정보를 제공한다. 현재 화이트리스트를 기반으로 웹사이트 보안수준에 대한 정보를 제공하는 솔루션은 거의 없으며, 이에 대한 자세한 정보는 전혀 제공되고 있지 않은 상황이다. 그러나 블랙리스트만을 기반으로 웹 사이트의 보안 수준 정보를 제공하는 것은, 피싱 사이트의 생존 시간이 길어야 3시간임을 감안할 때, 사고 발생 후에 적용된다는 면에서 비효율적이라 볼 수 있다.

그러므로 본 연구에서는 이러한 문제점을 해결하기 위하여 블랙리스트와 화이트리스트 모두를 기반으로 한, 웹 사이트의 보안수준에 대한 정보를 제공할 수 있는 시스템을 제안하고자 한다. 본 연구를 통해 개발하고자 하는 웹 사이트 보안 수준 확인 시스템은 웹 페이지의 물리적, 내부적, 외부적 검사를 통해 1차적으로 사이트에 존재하는 위험 요소를 검사해주고, 2차적으로 Website White List DB, Website Black List DB와의 연동 모듈을 통해 웹 사이트에 대한 신뢰도를 검사하여 개인 정보 침해 사고를 최소화 한다.

1차적으로 이루어지는 사이트의 물리적 분석은 접속 중인 웹 페이지의 URL 주소 분석을 통해 이루어지며, URL이 가지고 있을 수 있는 여러 가지

피싱 위협 요소 가능성을 점검한다. 내부적 분석은 페이지 내 HTML 소스를 분석함으로써 다른 사이트로의 링크 주소와 Input Type Pattern 분석을 통한 위험 요소 검출이 이루어진다. 사이트의 외부적 분석은 피싱 사이트에 사용되는 IP가 미국, 중국 등 해외 사이트가 많다는 점을 바탕으로, 접속 중인 페이지의 서버가 국내에 존재하는지의 여부를 가지고 판단한다.

2차적으로 이루어지는 신뢰도 검사에서의 Website Black List DB는 최신성을 유지하기 위해 날마다 PhishTank에서 XML문서로 제공하는 피싱 사이트 신고 목록을 업데이트 하였다. 또한 클라이언트 측에 설치된 Website Verification Mechnism 을 통해 접속하는 매 페이지마다 위험도 분석이 이루어져 위험성을 최소화 할 수 있다.

Ⅳ. 웹 시스템 내 위험성 분석 및 관련 메커니즘

접속 중인 사이트의 피싱 공격 존재 가능성을 분석하기 위해, 본 연구에서는 웹 페이지의 물리적, 내부적, 외부적 항목을 분석· 평가 한다. 피싱 사이트 여부를 가릴 수 있는 위험요소 중, 안전한 사이트에서도 검출 될 수 있는 판단 항목이 존재하기도 한다. 그렇기 때문에 각 위험 요소를 위험도별로 사용자에게 다르게 결과 정보를 제공한다.

페이지 위험도 측정은 매 페이지마다 이루어지는 검사로서 피싱 사이트에서 가장 많이 쓰이는 URL 변조의 존재 가능성을 검사해주고, 사용자의 민감한 정보를 요구하는지에 대한 검사를 한다. 안전한 사이트에서 사용되지 않는 요소가 검출 된 경우 사용자의 웹 브라우저 내에 위치한 툴바의 경고 표시를 이용하여 위험성을 알린다. 주민번호 입력창과 같이 안전한 사이트에서도 검출 될 수 있는 항목은 사용자에게 위험 요소 검출 사실을 알리고 보다 신뢰성 있는 신뢰도 검사 여부를 묻는 팝업 창을 보내게 된다. 팝업 메시지에 따른 사용자의 선택에 따라 2차적인 신뢰도 검사 실행 여부가 결정된다. 이번 장에서는 클라이언트 측 모듈에 적용하기 위한 웹 시스템 내의 물리적, 내부적, 외부적인 요인을 분석하여 보았다.

1. 물리적 분석 항목

사이트의 물리적 분석은 접속 중인 웹 페이지의 URL 주소 분석을 통해, URL이 가지고 있을 수 있는 여러 피싱 위협 요소 가능성을 점검함으로써 이루어진다. 위험 요소가 검출되어지는 결과에 따라 사용자측으로 이차적인 신뢰성 검사 여부를 묻는 팝업 메시지를 띄우는 것과, 툴바에 설치된 신호등을 통해 사이트의 신뢰도 결과를 전송한다. 다음은 본 시스템에서 점검하여 주는 URL에 있을 수 있는 위험 요소에 대한 정의 및 예시를 설명하도록 한 것이다.

1.1 IP 주소로 이루어진 URL

1.1.1 개요

URL의 도메인 네임 대신 IP주소를 사용하여 도메인 이름을 혼란스럽게 하여 콘텐츠 필터링을 피하거나 목적지를 속이기 위한 방식으로 IP주소를 Hex, Oct, 십진수 등 다양하게 나타낼 수 있다.

많은 수의 피싱 페이지 또는 사이트의 경우 IP 주소를 직접적으로 활용한 URL을 사용한다. 이러한 경우는 피싱 가능성이 상대적으로 적지만 사용자에게 피드백을 해야 하는 경우라고 판단된다.

정상적인 서비스를 제공하는 사이트의 경우, 특히 사용자의 중요한 정보와 연관성이 있는 사이트의 경우에 자사의 도메인을 이용하여 링크를 사용

하게 된다. 따라서 IP를 이용한 링크를 사용하면서, 사용자로부터 중요한 계좌 정보, 아이디 정보를 요구한다면, 이러한 사이트는 의심해 볼만한 사이트이다.

1.1.2 위험유형 및 사례분석

1) Paypal 사이트를 도메인 이름 대신 IP주소로 표현

IP로 구성된 URL정보를 가지고 있으면서, 디렉토리 이름을 www.paypal.com으로 구성함으로써 사용자들이 실제 www.paypal.com 사이트로 인식하도록 하고 있다.[20]

[그림 4-1] Paypal 피싱 사이트

2) IP 주소를 다양하게 표현하여 사용하는 방식

o 십진수 IP 주소 : http://210.134.161.35/
o Dword IP 주소로 표현 : http:// 3532038435/
o 8진수 IP 주소 표현 : http://0322.0206.0241.0043/
o HEX 주소 표현 : http://0xD2.0x86.0xA1.0x23/ 또는
http://0xD286A123/

1.2 Forged Domain

1.2.1 개요

실제 abc.com이란 도메인을 사용하면서, 사용자에게는 kbstar.com 과 같은 도메인으로 보이도록 사용자를 속이는 도메인 형태를 말한다. Forged Domain의 경우 대부분 3차 도메인 이상의 도메인 이름을 이용한다.

1.2.2 위험유형 및 사례분석

포털 사이트에서 링크되어 들어가는 쇼핑몰의 경우 URL에 최상위 도메인 2개 포함되는 것을 말한다. 하지만 네이버에서 링크되어 들어가는 쇼핑몰의 경우 URL에 최상위 도메인 2개 포함이 가능하여 이를 구별하는 알고리즘이 필요하다.

[그림 4-2] URL에 최상위 도메인 2개 포함

1.3 IDN Spoofing - ASCII Character Check

1.3.1 개요

IDN(Internationalized Domain Names)이란 영문자 대신 각 지역 언어의 문자를 그대로 사용하는 다국어 도메인 이름이다. IDN Spoofing은 거의 유사한 문자(예를 들면, 라틴어 또 는 로마자와 비슷한 키릴문자를 사용)를 사용하는 방식이다.

1.3.2 위험유형 및 사례분석

아래의 (그림 6-7)은 paypal.com의 문자 “a”를 키릴문자로 대체한 것이다. 거의 모양이 흡사 비슷하게 보이도록 한다.

http://www.p&#1072ypal.com (Original http://www.paypal.com)
→ http://www.paypal.com

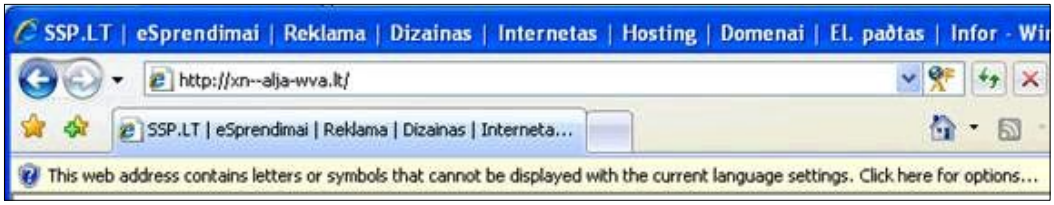
[그림 4-3] IDN Spoofing 사례 1

밑줄로 표시된 문자는 알파벳 ‘o’가 아니라 키릴문자 ‘o’이다.

http://www.theshmoоgroup.com/
(http://www.theshmoogroup.com)
→ http://www.theshmoogroup.com/

[그림 4-4] IDN Spoofing 사례 2

ASCII코드 및 Unicode 문자열을 DNS에서 문자셋을 다른 문자(키릴 문자)로 맵핑되지 않도록 제한해야 한다.(Punycode 사용)



[그림 4-5] 키릴문자를 사용한 도메인

1.4 특수 기호 사용 URL

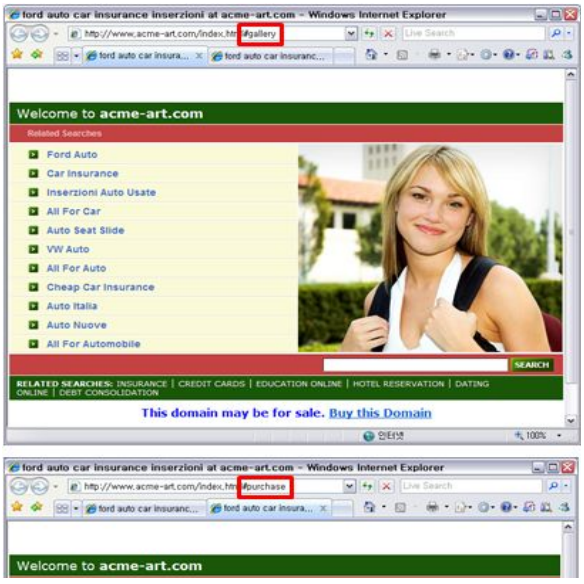
1.4.1 개요

대부분 URL 문자열은 문자, 숫자, 그리고 URL 문자열에 포함되면 특별한 의미를 갖는 예약된 특수 문자로 구성된다. 몇몇 URL 문자열에서 발견되는 다른 특수 문자들은 URL에 관한 한 특별한 의미를 갖지 않는다. 하지만, 이런 특수 문자들은 웹 서버를 통해 요청 받는 URL 또는 애플리케이션을 받아들이는 웹 서버를 위한 특별한 의미를 가지고 있다.

‘*’, ‘;’, ‘|’, ‘`’ 와 같은 문자들은 애플리케이션과 스크립트에서 메타 문자들로서 특별한 의미를 갖는다. 이러한 문자들은 URL에 어떠한 영향도 미치지 않는다. 하지만 마지막으로 그들이 애플리케이션에 전달이 된다면 완전히 입력의 의미가 달라질 것이고 때로는 보안상의 구멍을 만들 것이다.

1.4.2 위험유형 및 사례분석

'#' : 웹 페이지 안에서 어떠한 지점을 표시할 때 사용된다. 아래의 [그림 4-6]은 ‘#’가 사용되는 예시이다.



[그림 4-6] 특수기호 ‘#’을 이용한 페이지

1.5 비정상적으로 긴 도메인 주소

1.5.1 개요

Microsoft Internet Explorer는 최대 2,083문자의 Uniform Resource Locator(URL) 길이를 갖는다. 이 제한은 POST request와 GET request 양쪽에 모두 URL 길이에 적용된다.

1.5.2 위험유형 및 사례분석

아래의 [그림 4-7]은 최상위 도메인이 나오기까지의 도메인 네임이 지나치게 긴 예시 사이트이다.

[그림 4-7] 비정상적으로 긴 도메인

1.6 Keyword 공격

- URL Redirection, Bad Domain Name, Friendly Login URL

1.6.1 개요

1) URL Redirection 기능 이용

URL를 조작하는 피싱 공격 방법 중 URL Redirection을 활용하는 방법도 많이 사용된다. URL Redirection기법 중에서도 구글 검색엔진의 Redirection 기능을 이용하여 Long URL 형태로 공격하는 형태가 일반적이다.[21]

2) Bad Domain Name

피해자가 접속하기 위한 위조 사이트의 도메인을 실제사이트와 유사하게 만들어 피해자로 하여금 도메인 구분이 어렵도록 하여 피해자를 속이는 방법이다.

3) Friendly Login URL

Friendly login URL은 사용자들이 방문하고 있는 사이트가 인증된 페이지인 것처럼 URL을 조작할 수 있다.

1.6.2 위험유형 및 사례분석

1) URL Redirection 기능 이용

o URL Redirection 기능을 이용하여 피싱 사이트로 유도하는 방식

- http://www.google.com/pagead/iclk?sa=l&ai=x&adurl=http://www.hacker.com

아래의 사례는 URL Redirection를 이용한 Phishing의 경우이다.

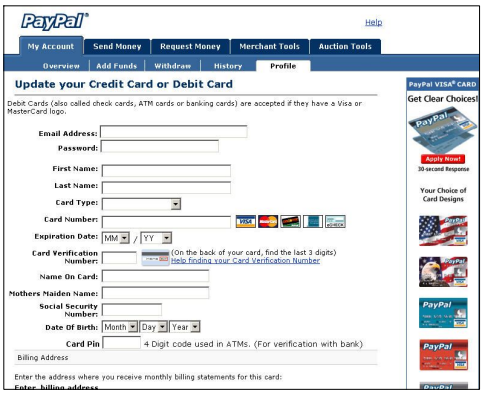
o Visible Link :

http://www.citizensbankonline.com/logon/securesurvey.asp

o Actual Link : http://review-data.org/go.html

o Phishing Link :

http://83.16.123.18/icons/pp/update.htm?=https://www.paypal.com/=cmd_login_access_account_up



[그림 4-8] URL Redirection을 이용한 Phishing

2) Bad Domain Name

o 유사 URL를 이용하여 사용자가 혼동하도록 하는 방법

정상적인 http://www.mybank.com/ 대신에 http://www.mybank.com.ch처럼 유사형태의 도메인을 악용하는 방법이다.

o Citizens Bank 도메인 위조

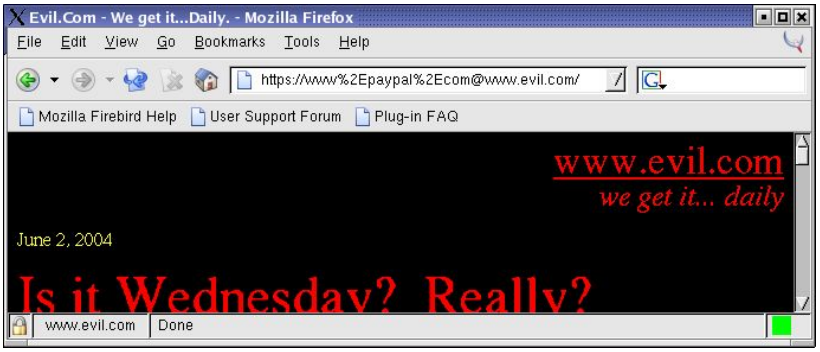
[그림 4-9] 도메인 위조 웹 사이트

위의 예에서 공격자는 http://www.citizensbankonline.com 을 클릭 하였으나 실제 접속된 곳은 http://www.citizenscolorsbankonline.com/default 로 도메인 이름이 교묘하게 다른 것을 알 수 있다.

3) Friendly Login URL

o 일반적인 피싱 기법으로 인증한 정보처럼 위장하여 URL를 조작 하는 방법

- https://www%2Epaypal%2Ecom@www.evil.com/



[그림 4-10] www.evil.com의 URL 조작

2. 내부적 분석 항목

웹 사이트의 내부적 분석은 페이지 내 HTML 소스를 분석함으로써 이루어진다. 내부적 분석을 통해 잡아주고자 하는 위험요소는 크게 두 가지 이다.

첫째로 많은 피싱 사이트로의 접속이 메일이나 다른 사이트에서의 링크를 통해 이루어진다는 점을 고려하여 HTML 소스의 Link Tag를 분석한다. 이를 통해 실제 Link 주소와 표기되는 주소를 속이는 방법을 사용하고 있는지 검사한다.

두 번째는 페이지 내의 Input Type Pattern을 분석하여 사용자에게 민감한 정보를 요구하는 입력 창이 있다면 사용자에게 이를 알려 주의를 하도록 한다.

2.1 Input Type Pattern

2.1.1 개요

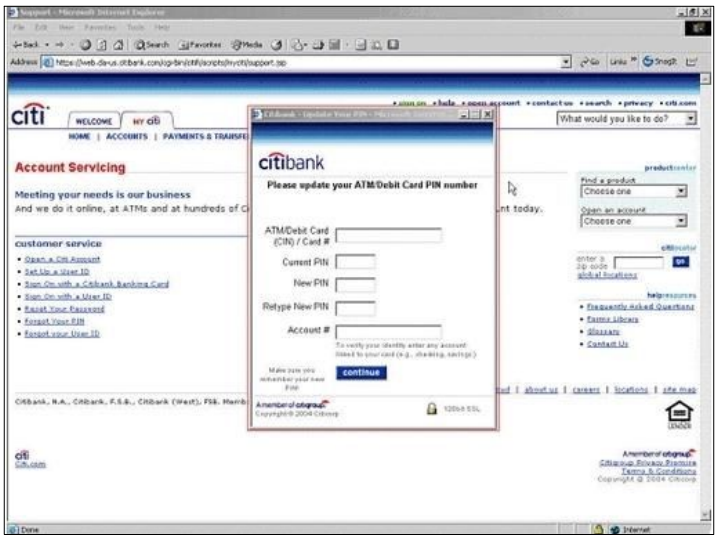
현재 대부분의 사이트는 서비스 이용 시 회원가입을 해야 한다. 개인 정보를 요구하는 사이트들 중 금융정보를 다루는 사이트는 일반 개인 정보를 요구하는 사이트 보다 좀 더 높은 수준의 보안을 요구한다.

2.1.2 위험유형 및 사례분석

o 개인정보 입력요구

[그림 4-11] 메일로 개인정보 입력요구

o 금융정보 입력요구



[그림 4-12] 금융정보 입력요구 창

2.2 URL Address Spoofing (%00, %01)

2.2.1 개요

대부분 URL Address Spoofing 기법은 Microsoft의 Internet Explorer의 취약점을 이용 한 것이 가장 많다. 예를 들면 아래와 같은 IE의 %01, %00 처리를 못해서 발생하는 취약점으로 인해 Windows 상태 창에서 클릭 시 피싱 사이트 URL이 보이지 않게 할 수 있는 해킹 공격이 가능하다.[22]

2.2.2 위험유형 및 사례분석

웹사이트에 http(s)://username:password@server 와 같이 기본인증을 통해 접근하는 방법은 웹브라우저의 주소창에 잘못된 URL을 표시하는 취약점이 존재한다.

이 취약성은 특정한 웹브라우저 버전에 대한 취약성으로 해당 취약점은 URL링크의 소스파일에 %01, %00 또는 @가 들어가 있으면 %01, %00 또는 @ 이하의 주소는 표시하지 못하여 제대로 된 URL을 표시하지 못하게 된다.

따라서 사용자가 클릭한 링크와 다른 사이트를 방문하여도 주소창에서 확인이 불가능하여 사용자는 위조 사이트를 신뢰된 사이트로 믿게 된다. 웹브라우저가 해당 취약점에 취약한지의 여부는 다음사이트에서 테스트해볼 수 있다.

```
<br>
<b>Click this link to see if you are vulnerable:</b><br>
<a href="http://www.microsoft.com%00@secunia.com/
internet_explorer_address_bar_spoofing_test/" style="font: 8pt verdana, sans-
serif;">
Click Here to Perform Test!
</a>
<br>
```

[그림 4-13] 소스코드 내 삽입된 취약한 특수문자

소스코드 내에 문자 %00가 존재하는 것을 확인할 수 있다.

[그림 4-14] 취약한 웹브라우저의 주소창1

위와 같이 해당 취약한 웹브라우저를 사용할 경우 %00@ 이하의 URL 주소는 표시가 되지 않는다.

취약한 브라우저에서 해당 링크를 클릭하였을 경우 주소표시 창에 표시되는 것은 http://www.microsoft.com이나 실제 접속하는 사이트는 http://secunia.com/이 된다. 즉 사용자가 특정 링크를 클릭할 경우 원치 않는 페이지로 접근될 수 있다. 취약한 브라우저는 업데이트를 하여 http(s)://username:password@server 와 같은 기본인증의 사용을 중지시켜야 한다.

[그림 4-15] 취약한 웹브라우저의 주소창2

2.3 href link(URL) Check

- IDN Spoofing-ASCII , 유니코드 인코딩

2.3.1 개요

가장 일반적으로 사용하는 피싱 방식 중 하나로 유니코드 형태로 URL을 표현한다. 유니코드에서는 다른 문자의 부분에 어떤 ASCII 바이트도 나타낼 수 없고, 이론적으로 6바이트 길이까지 가능하다.[23]

2.3.2 위험유형 및 사례분석

o '이OO'을 '%EC%9D%B4%EB%AA%85%EB%B0%95'로 표현

[그림 4-16] UNICODE로 인코딩된 URL

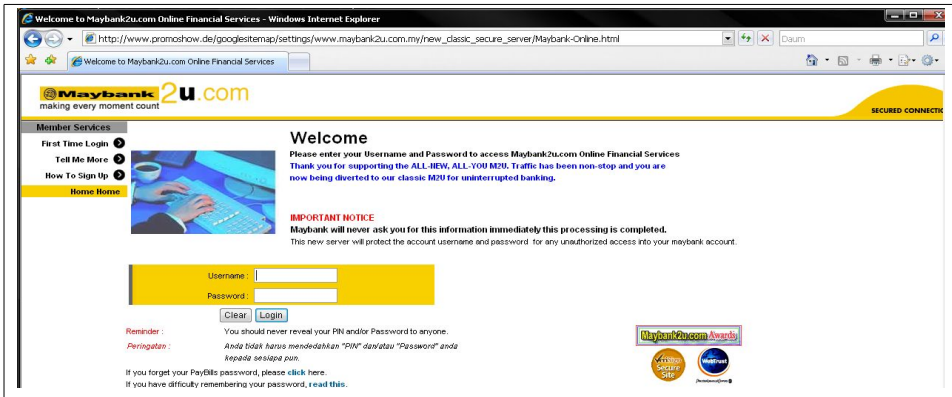
2.4 ISO-8859-1 인코딩

2.4.1 개요

ISO-8859-1은 서유럽 언어의 표기에 필요한 US-ASCII에 없는 94개의 글자의 순차적 나열이다.

ISO-8859-1은 퓨니코드(ex.키릴문자)를 사용하여 도메인에 모든 유니코드의 문자를 쓸 수 있기 때문에 IDSN을 사용하여 피싱 공격을 할 수 있다.

2.4.2 위험 요소 및 유형 분석



[그림 4-17] ISO-8859-1이 사용된 피싱 사이트 예시

3. 외부적 분석 항목

3.1 Nation Check

3.1.1 개요

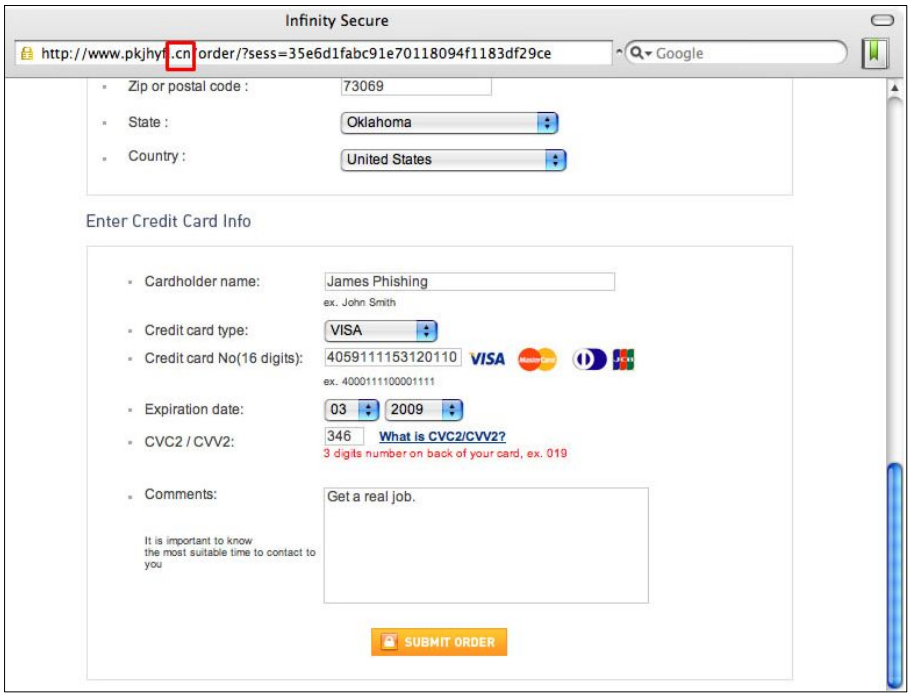
2007 시만텍에서 발표한 월간 피싱 보고서에 따르면 피싱 사이트에 사용되는 IP는 미국이 가장 많고, 공격 대상이 되는 사이트도 ebay나 paypal, amazon 등 해외의 주요 사이트들이 많이 사용된다. 또한 우리나라를 공격 대상으로 하여 만들어진 피싱 사이트는 대부분 중국에 IP를 두고 있다.

피싱 경유지로 이용된 시스템들은 해킹을 당했거나 노출된 취약점들로 인하여 DDOS공격, 스팸릴레이, 악성코드유포 등 각종 다른 보안 사고에도 악용될 수 있어 다른 시스템이나 네트워크에 피해를 줄 수 있다.

또한 경유지 사이트에 대한 삭제 및 시스템 보안조치 등이 적절하게 이뤄지지 않을 경우 해당 시스템이 속한 IP대역이 블랙리스트로 차단함으로써, 특정 사이트의 접속이나 메일발송 등이 제한할 수 있다.

3.1.2 위험유형 및 사례분석

대부분의 피싱 사이트의 진원지가 해외, 특히 미국, 중국임을 감안하여 접속 중인 페이지의 IP 주소의 범위를 분석하여 해외 발 사이트인 경우 사용자에게 경고를 주어 웹 페이지 이용에 더 주의 할 수 있도록 한다.



[그림 4-18] 서버가 중국에 있는 웹 페이지

Ⅴ. 시스템 설계 및 구현

1. 시스템 기본 구조 및 시나리오

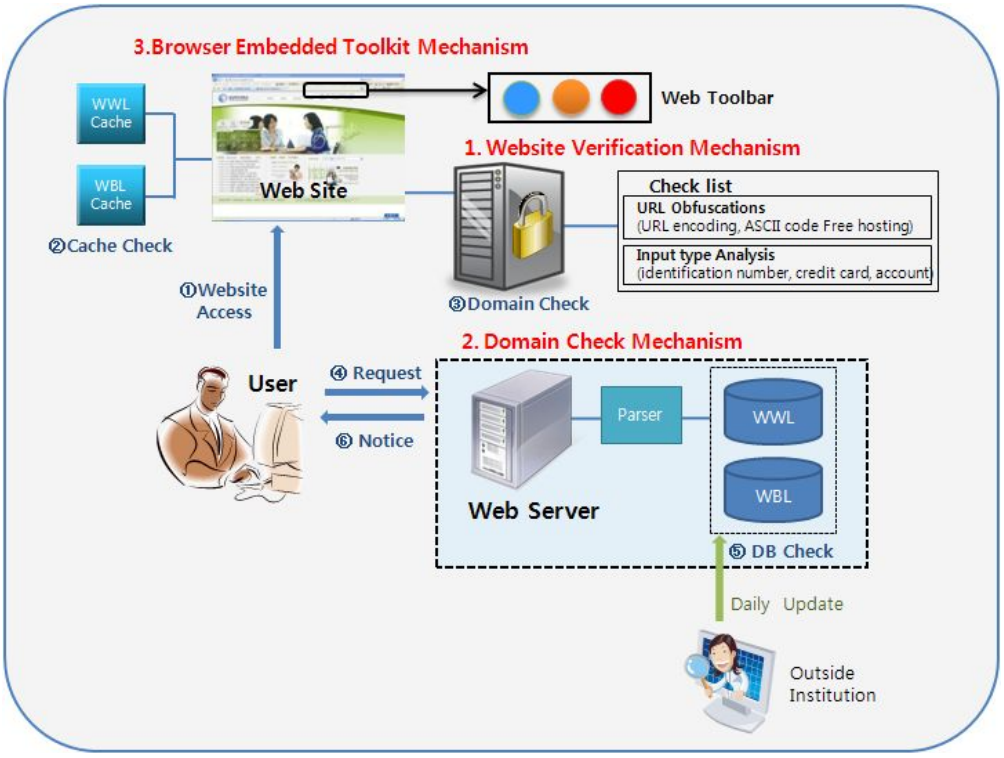
1.1 요구사항 분석

본 연구에서는 웹 환경에서의 개인정보 공격에 대한 효율적인 대응 방안으로 WVMS(Website Verification Management System)을 제안하여 보았다. WVMS에서는 기본적으로 WWL과 WBL DB를 갖고 있는 서버와 웹 시스템의 위험을 판단하고 사용자에게 피드백을 해 줄 수 있는 클라이언트가 필요하며, 구현 단계에서 다음과 같은 고려 사항이 필요하다.

① 클라이언트 측 고려 사항
- 사용자가 요청하는 웹 페이지에 대한 정보의 확인이 가능해야 한다.
- 사용자에게 피드백하기 용이해야 한다.
- 사용자가 클라이언트 프로그램으로 인한 불편(속도 등)이 최소화 되어야 한다.

② 서버 측 고려 사항
- 많은 수의 클라이언트의 요청을 효율적으로 처리해야 한다.
- 서버 자체에 대한 보안을 강화해야 한다.

1.2 보안 위험도 확인 시나리오



[그림 5-1] 시스템 기본 구조

① 사용자는 인터넷에 접속하기 위하여 웹 브라우저에 URL을 입력한다.

② 입력된 URL에 대해 클라이언트 측에서 캐시를 검색한다. 캐시에 정보가 있는 경우 WWL, WBL에 따라 툴바의 신호등 불빛을 통해 정보를 바로 제공한다.

③ 캐시에 없는 경우 클라이언트 측에 설치된 Website Verification Mechanism을 통해 URL 변조 공격과 Input Type Pattern을 분석한다. 분

석결과에 따라 위험요소가 검출 되는 경우 바로 신호등을 통해 알려준다.

④ 앞의 과정 ③에서 검출되는 위험 요소에 따라 사이트 내 의심되는 요소가 발견되는 경우 사용자에게 신뢰성 검사 여부를 묻고 이에 대한 사용자의 의견 선택에 따라 본 시스템의 서버에 연결된 WWL, WBL DB를 검색하게 된다. 검색 결과에 따라 신호등을 통해 알려준다.

2. 시스템 상세 설계

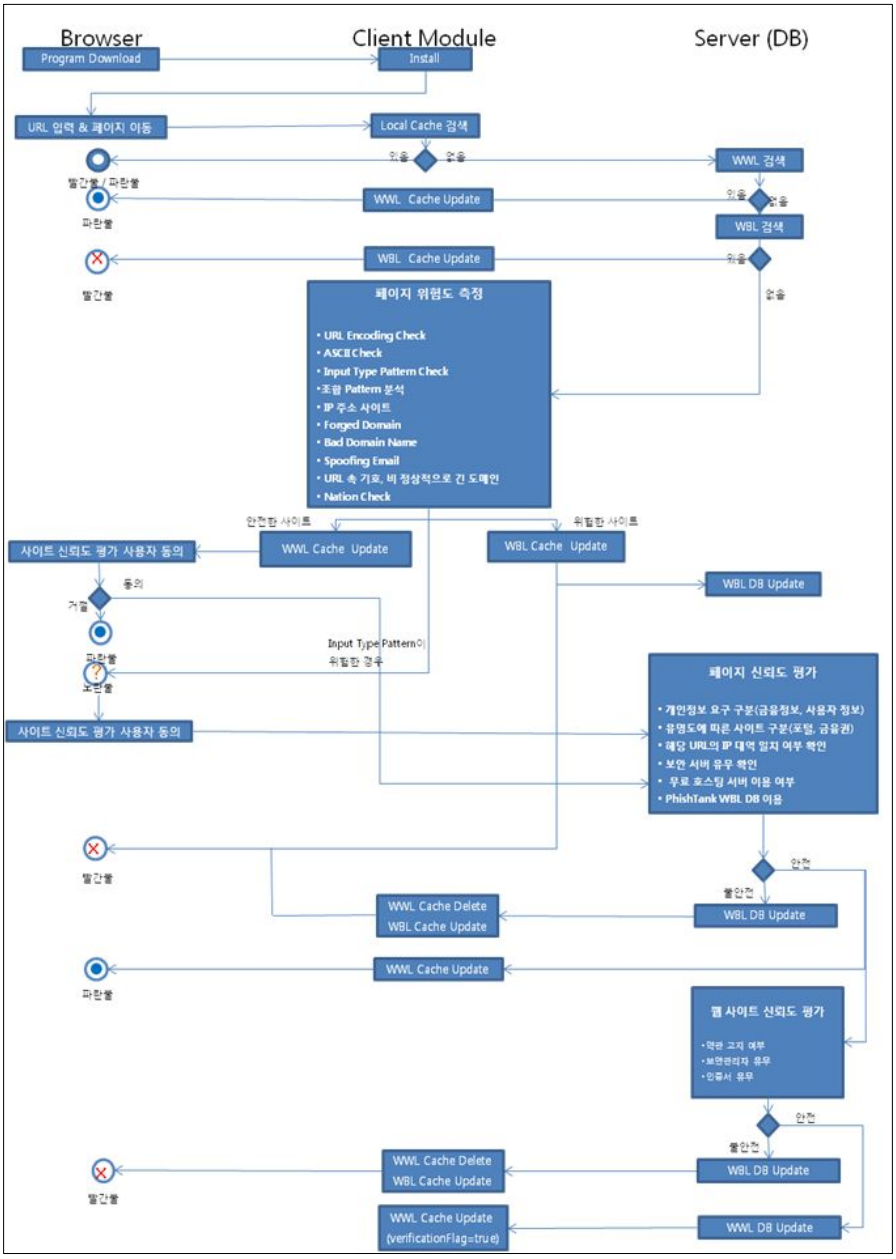
2.1 클라이언트

클라이언트는 웹 브라우저의 플러그인 형태로 설치되며 툴바 위치에 삽입된다. 클라이언트 시스템은 사용자의 요청으로부터 발생한 웹 서버 응답 페이지를 분석하여 WWL, WBL DB와의 비교, 위험도 측정 단계를 통한 결과를 사용자에게 '안전', '경고', '위험'의 세 단계로 표시하게 된다.

사용자가 요청하는 웹 페이지의 빠른 신뢰성 평가를 위해 WWL, WBL Local Cache를 갖게 되며 이는 사용자가 웹 서핑 시 일정한 개수의 사이트를 서핑 한다는 점을 이용한 것이다. 빈도수가 높은 안전한 사이트는 WWL Cache에 저장하고, 웹 서핑 시 WBL DB에 있던 사이트를 방문, 혹은 사이트 신뢰도 평가 시 '위험' 평가를 받은 사이트는 WBL Cache에 저장하여 차후 접속 시 빠른 응답을 받을 수 있도록 한다. 또한 이 Cache에 저장된 정보는 타임스탬프(Timestamp)와 비교하여 항상 최신성을 유지하도록 한다.

사용자가 웹 서핑 시 URL을 입력하거나 페이지 이동 시 BHO와 레지스트리를 이용하여 URL과 IP를 추출하게 된다. 이를 통해 이전 페이지와의 IP 일치 여부를 확인한 후, 일치하면 Local Cache와 DB 검색을 하지 않고 매 페이지마다 피싱 공격의 가능성을 검사해준다. 이전 IP와 일치하지 않은 경우에는 Local Cache와 DB검색을 하여 WWL, WBL 검색 결과에 따른 신호등을 보여준다. 그 이후에도 피싱 공격의 위험성은 '페이지 위험도 검사'를 통해 Client Module에서 매 페이지마다 이루어진다.

'페이지 위험도 검사'는 URL위조 검사와 Input Box의 Pattern 분석을 하게 되는데, URL위조는 어떤 경우에도 위험하다고 판단하여 발견 즉시 사용자에게 경고를 보낸다. 또한 Input Box의 Pattern이 개인의 민감한 정보를 요구하는 경우, 팝업 메시지를 통해 보다 심도 있는 '페이지 신뢰도 검사' 실시 여부에 대한 의견을 묻는다. URL 위조와 의심스러운 Input Box가 없는 경우에도 사용자에게 '페이지 신뢰도 검사' 실시 여부에 대한 의견을 묻는다. 사용자가 동의한 '페이지 신뢰도 검사'는 WVMS의 Server에서 이루어지게 된다. 이러한 일련의 과정은 아래의 [그림 5-2]에 정리되어 있다.


Browser
Client Module
Server (DB)
Program Download
Install
URL 입력 & 페이지 이동
Local Cache 검색
있음
없음
WWL 검색
발견됨 / 파란색
WWL Cache Update
있음
없음
파란색
WBL 검색
WBL Cache Update
있음
발견됨
없음
페이지 위험도 측정
• URL Encoding Check
• ASCII Check
• Input Type Pattern Check
•조합 Pattern 분석
• IP 주소 사이트
• Forged Domain
• Bad Domain Name
• Spoofing Email
• URL 속 기호, 비 정상적으로 긴 도메인
• Nation Check
안전한 사이트
위험한 사이트
사이트 신뢰도 평가 사용자 동의
WWL Cache Update
WBL Cache Update
WBL DB Update
동의
거절
파란색
Input Type Pattern이
위험한 경우
보통색
페이지 신뢰도 평가
• 개인정보 요구 구분(금융정보, 사용자 정보)
• 유명도에 따른 사이트 구분(포털, 금융권)
• 해당 URL의 IP 대역 일치 여부 확인
• 보안 서버 유무 확인
• 무료 호스팅 서버 이용 여부
• PhishTank WBL DB 이용
사이트 신뢰도 평가 사용자 동의
발견됨
안전
불안전
WWL Cache Delete
WBL Cache Update
WBL DB Update
WWL Cache Update
파란색
웹 사이트 신뢰도 평가
•악관 고지 여부
•보안관리자 유무
•인증서 유무
안전
불안전
WWL Cache Delete
WBL Cache Update
WBL DB Update
발견됨
WWL Cache Update
(verificationFlag=true)
WWL DB Update

[그림 5-2] 시스템 설계도

이를 토대로 제안하는 시스템의 알고리즘은 다음과 같이 간략화 할 수 있다.

```
main()
{
        url = get URL;
        ip = get IP;
        serverConnection();
        if(ip == previousIp)
                calculateRiskLevel();
        else
                localCacheCheck();
}
void calculateRiskLevel()
{
        color="gray";   // 테스트 되지 않은 페이지는 알 수 없음으로 표시
        URLEncodingCheck();
        ASCIICodeCheck();
        tagCheck();
        nationCheck();
        if(color=="red" && color!="yellow")
        {
                localWWLCacheUpdate();
                verificationFlag=flase;
                status="현재 접속한 웹페이지는 위험이 발견되지 않았습니다.
                        보다 안전한 측정을 위해 신뢰성 검사를 하시겠습니까?" ;
         MessgeBox( NULL, status, "사용자동의", MB_OK );
         if(clientAgree== true)
                        serverRequestTrustLevel();
                else
               color="green";
    }
        if(color=="red")
        {
                localWBLCacheUpdate();
                server.WBLDBUpdate();
        }
        if(color=="yellow" && clientAgree== true)
                serverRequestTrustLevel();
    sendSignal();
}
void serverRequestTrustLevel()
{
        // 서버에 웹페이지 신뢰도 측정 요청
```

```
        if(server.pageTrustLevel()==true)
    {
                localWWLCacheUpdate();
                color="green";
                // 서버에 웹사이트 신뢰도 측정 요청
                if(server.decideSiteTrustLevel()==true)
                {
                        verificationFlag=true;
                        server.WWLDBUpdate();
                }
                else
         {
                        localWWLCacheDelete();
                        localWBLCacheUpdate();
                        server.WBLDBUpdate();
                        color="red";
         }
    }
        else
    {
                 localWBLCacheUpdate();
                 server.WBLDBUpdate();
          color="red";
    }
}
void localCacheCheck()
{
        if(localWWLCacheCheck()==true)
                color="green";
        else
        {
                if(localWBLCacheCheck()==true)
                        color="red";
                else
                {
                        if(server.WWLCheck()==true)
                                color="green";
                        else
                        {
                                if(server.WBLCheck()==true)
                                        color="red";
                                else
                                        calculateRiskLevel();
                        }
                }
```

```
        }sendSignal(); // toolbar에 페이지 위험 수준 전송(신호등)
}
void URLEncodingCheck(char url)
{
        char* token = strtok( url, "%" );// 문자열을 '%' 단위로 분리
        if( token )
        {
                value = GetBytes(2);            // 분리자 이후 2자리 추출
                if(value == CharacterCode)
                        URLAttack = false;
                else
                {
                        URLAttack = true;
                        color="red";
                        sendSignal();
                        server.WBLDBUpload();
                }
        }
}
void ASCIICodeCheck(char url)
{
        char url[Length];
        for(i=0; i<url.length; i++)
                url[i] = url;
        for(i=0; url[i]!=0 ; i++)
        {
                int a = (int)url[i];
                if(a<0 || 127<a)        //변환된 숫자가 ASCII 범위인지 검사
                {
                        color="red";
                        sendSignal();
                        server.WBLDBUpload();
                        localWWLCacheUpdate();
                }
                else
                        URLAttack = false;
        }
}
void tagCheck()
{
        if (sTempTagName == _T ("input")) // tag 값이 input인 경우
        {
                // text box의 패턴이 주민번호, 아이핀, 계좌·카드번호인 경우
                if(sTempType == _T ("text") && patternCheck()==true)
                        privacyInputText += 1;
```

```
                if(sTempType == _T ("submit") || sTempType == _T ("image"))
                        privacySubmitButton = true;
        }
        // 개인정보입력을 요구한경우 Client에게 Notice
        if(privacyCheck>0 && privacySubmitButton==true)
        {
                status="현재 접속한 웹페이지는 개인정보 노출의 위험이 있습니다.
                        보다 안전한 측정을 위해 신뢰성 검사를 하시겠습니까?" ;
                color="yellow";
                MessgeBox( NULL, status, "사용자동의", MB_OK );
                sendSignal();
        }
}
void NationCorrectness(char ip)
{
        if(KoreaIpRangeMin < ip < KoreaIpRangeMax)
                ipNation = "korea";
        else
        {
                ipNation = "abroad";
                status="현재 접속한 사이트는 외국사이트입니다.";
                color="yellow";
                sendMessage();
                sendSignal();
        }
}
```

[그림 5-3] WVMS Client Module 알고리즘

2.2 서버

Client 측에서 보내진 사이트의 정보로 WVMS Server에서 DB를 검색한다. WWL DB, WBL DB 두 가지로 분류된 이 DB는 관리자에 의해 미리 저장 되어 있거나, 사이트 신뢰도 평가 후 '안전' 평가 등급을 받은 사이트는 WWL DB에 '위험' 평가 등급을 받은 사이트는 WBL DB에 자동으로 등록되게 된다. 본 시스템에서의 DB 이용은 Cache에 없는 모든 사이트에 대해 신뢰도 평가를 하지 않아도 되도록 하여 평가 결과를 사용자에게 좀 더

빠른 속도로 알려질 수 있도록 한다.

또한 '웹 페이지 위험도 측정' 이후의 '페이지 신뢰도 검사' 검사는 이전의 검사에서 의심스러운 Input Type이 발견되는 경우, 혹은 사용자에게 더 깊은 신뢰도를 주기 위해 이루어지는데 평가 항목으로는 Input Type의 개인정보 요구 사항과 사이트의 신뢰도 및 유명도, 해당 URL과 IP의 일치 여부, 보안 서버의 유무를 확인하게 된다. 이를 통해 페이지의 신뢰도에 대해 심도 있는 평가를 하게 되고, 평가 항목에서 위험 요소가 발견되면 WBL DB와 Local Cache에 저장되게 된다. 위험 요소가 발견 되지 않은 경우에는 '웹 사이트 신뢰도 평가'를 통해 WWL DB에 Update할 수 있는 자격이 되는지 최종적으로 검사한다. 이 검사는 사이트의 Security Compliance등 사이트의 보안성에 대한 강화된 평가를 하여 WWL DB의 신뢰성을 유지하도록 한다.

이를 토대로 제안하는 시스템의 알고리즘은 아래 [그림 5-4]와 같이 간략화 할 수 있다.

```
main()
{
        serverActivation();
        clientConnection();
}
bool WWLCheck(char url)
{
        CString query_statement = "select * from WWL", insert_string;
        RETCODE result = SQLExecDirect(h_statement, (unsigned char *)
              (const char *)query_statement, SQL_NTS);
        if(result == SQL_NO_DATA_FOUND)
        {
              status="WWL DB에존재하지않음";
              sendMessage();
```

```
                return false;
        }
        else
                return true;
}
bool WBLCheck(char url)
{
        CString query_statement = "select * from WBL", insert_string;
        RETCODE result = SQLExecDirect(statement, (unsigned char *)
              (const char *)query_statement, SQL_NTS);
        if(result == SQL_NO_DATA_FOUND)
        {
                status="WBL DB에 존재 하지 않음";
                sendMessage();
                return false;
        }
        else
                return true;
}
void WWLDBUpdate(char url)
{
        if(MODE == add)          // 추가모드
        {
                str.Format("Insert into WWL values (\%d, %d, %d, %d, %d, %d, %d)",
                domain,site_name,site_type,nation,security_compliance,mainip,subip);
        }
        else              // 변경모드
        {
                str.Format("Update WWL Set Domain = %d, Site Name= %d, Site Type = %d,
         Nation = %d, Security Compliance= %d, Main IP = %d, Sub IP = %d Where domain
         = %d",
                domain,site_name,site_type,nation,security_compliance,mainip,subip,domain);
        }
        SQLExecDirect(statement, (unsigned char *)(const char *)str, SQL_NTS );
        SQLTransact(mh_environment, statement, SQL_COMMIT);
        SQLFreeStmt(statement, SQL_DROP);
}
void WBLDBUpdate(char url)
{
        if(MODE == add)          // 추가모드
        {
                str.Format("Insert into WBL values (\%d, %d, %d, %d, %d, %d)",
                        domain, nation, security_compliance, mainip, subip, attack_date);
        }
        else              // 변경모드
```

```
		{
			str.Format("Update WBL Set Domain = %d, Nation = %d, SecurityCompliance
		 = %d, Main IP = %d, Sub IP = %d, Attack Date = %d  Where domain = %d",
			domain, nation, security_compliance, mainip, subip, attack_date, domain);
		}
		SQLExecDirect(statement, (unsigned char *)(const char *)str, SQL_NTS );
		SQLTransact(mh_environment, statement, SQL_COMMIT);
		SQLFreeStmt(statement, SQL_DROP);
}
void WWLDBDelete(char url)
{
		str.Format("delete from WWL where domain = %d, domain);
		SQLExecDirect(statement, (unsigned char *)(const char *)str, SQL_NTS );
		SQLTransact(mh_environment, statement, SQL_COMMIT);
		SQLFreeStmt(statement, SQL_DROP);
}
pageTrustLevel(char url, char ip) //웹페이지 신뢰도 평가
{
		if(WWLCheck(url)==true)
		{
			CString query_statement = "select site_type from WWL", insert_string;
			RETCODE site_type_result = SQLExecDirect(statement, (unsigned char *)
		 (const char *)query_statement, SQL_NTS);
			CString query_statement = "select subip from WWL", insert_string;
			RETCODE ip_result = SQLExecDirect(statement, (unsigned char *)
		 (const char *)query_statement, SQL_NTS);
			CString query_statement = "select security_compliance from WWL",
			insert_string;
			RETCODE security_compliance_result = SQLExecDirect(statement,
		 (unsigned char *)(const char *)query_statement, SQL_NTS);
		}
		if(WBLCheck(url)==true)
		{
			CString query_statement = "select site_type from WBL", insert_string;
			RETCODE result = SQLExecDirect(statement, (unsigned char *)
		 (const char *)query_statement, SQL_NTS);
			CString query_statement = "select subip from WBL", insert_string;
			RETCODE ip_result = SQLExecDirect(statement, (unsigned char *)
		 (const char *)query_statement, SQL_NTS);
			CString query_statement = "select security_compliance from WWL",
			insert_string;
			RETCODE security_compliance_result = SQLExecDirect(statement,
		 (unsigned char *)(const char *)query_statement, SQL_NTS);
		}
		// 유명도에 따른 사이트 구분(포털, 금융권)
```

```
        switch(site_type_result)
        {
        case free:                          //무료호스팅업체
                total = total+(3*4);
        case newnon-banking sector:    //제3 금융권
                total = total+(3*3);
        case portal:                         //포털사이트
                total = total+(3*2);
        case government:                  //정부
                total = total+(3*1);
        case non-banking sector:         //제2 금융권
                total = total+(3*0);
        case banking sector:               //제1 금융권
                total = total+(3*0);
        }
        // 해당 URL의 IP 대역 일치 여부 확인
        if(ip != ip_result)
                total = total+(5*4);
        // 보안 서버 유무 확인
        if(security_compliance_result==false)
                total = total+(4*4);
}
```

[그림 5-4] WVMS Server 알고리즘

2.3 데이터베이스

2.3.1 WWL DB 구축 범위

본 연구에서는 사용자들의 서비스 이용도가 높고, 피싱 공격의 대상이 되는 사이트를 중심으로 WWL DB를 구축하고자 한다.

피싱 공격의 주된 방법 중 하나가 이메일을 이용한 공격이고, 인터넷 이용 시 많은 사람들이 이용하는 기본적인 서비스인 점을 고려하여 주로 이용되는 이메일 사이트를 DB에 저장하여 응답시간을 최소화하고자 한다. 또한

이메일 서비스를 제공하는 사이트(주로 포탈)에서 금융정보를 묻는 것은 위험한 경우로 판단하여 이를 막고자 함에 있다.

피싱 사이트의 가장 큰 범위를 차지하는 금융권의 사이트를 저장하여 이외의 사이트에서 금융정보를 요구하는 일을 막아주도록 한다. 금융권을 제1, 제2, 제3 금융권별로 나누어 보안이 비교적 약하여 해킹 및 공격의 위험성이 높은 제 3금융권은 신뢰도 평가 시 차이를 주도록 한다.

[표 5-1] WWL DB에 사용될 사이트 목록 예시

E-mail	금융권		
	제1금융권	제2금융권	제3금융권
www.chol.com www.dreamwiz.com www.empas.com www.hanmir.com www.freechal.com www.hanmail.net www.orgio.com www.naver.com www.nate.com www.lycos.co.kr www.korea.com www.kebi.com www.hotmail.com www.hanafos.com www.yahoo.com www.yahoo.co.kr www.paran.com	**특수은행** 한국은행, 한국산업은행, 한국수출입은행, 농업협동조합중앙회 수산업협동조합중앙회 **일반은행** 국민은행, 신한은행, 제일은행, 우리은행, 하나은행, 기업은행, 외환은행, 산업은행, 씨티은행 **지방은행** 부산은행, 대구은행, 광주은행, 제주은행, 전북은행, 경남은행	**보험사** 삼성생명, 동양생명, 대한생명, 교보생명, 금호생명 삼성화재해상보험, 현대해상화재보험, 쌍용화재해상보험, 신동아화재해상보험, 엘지화재해상보험, 대한화재해상보험, 동부화재해상보험, 동양화재해상보험 **카드사** 외환카드, 현대카드, 동양카드, 롯데카드 **캐피탈사** 현대캐피탈, 국민캐피탈, 효성캐피탈, 금호캐피탈, 산은캐	산와머니 러시앤캐쉬 원캐싱 이지캐피탈 미즈사랑 등

		피탈, 한국캐피탈, 연합캐피탈, 썬캐피탈, 동부캐피탈, 롯데캐피탈 **투자신탁** 굿모닝증권, 서울증권, 대우증권, 동원증권, 동부증권, LG투자증권 **저축은행** 부산상호저축은행, 흥국상호저축은행, 경은상호저축은행, 강원상호저축은행, 하나로상호저축은행, 한마음상호저축은행, 으뜸상호저축은행, 동원상호저축은행, 한중상호저축은행, HK상호저축은행, 솔로몬저축은행, 한국저축은행, 현대스위스저축은행	

2.3.2 WBL DB 구축 및 Update

WBL DB의 내용은 공신력 있는 해외의 사이트(Phishtank)에서 제공하는 피싱 공격 사이트 목록을 이용한다. 사이트 목록은 XML 문서로 무료로 제공되고 날마다 업데이트되기 때문에 효율적으로 피싱 사이트를 막을 수 있다.

아래의 [그림 5-5]는 Phishtank에서 제공하는 XML 문서의 예시이고, 이 XML 문서를 이용하여 DB에 데이터를 저장하는 알고리즘은 [그림 5-6]이다.

```
<?xml version="1.0" encoding="utf-8"?>
 <output>
  <meta>
   <generated_at>2008-08-06T07:51:40+00:00</generated_at>
   <total_entries>4155</total_entries>
  </meta>
  <entries>
   <entry>
    <url><![CDATA[http://gmglory.com/cns/abbey.com/?usid=PhishTank_was_here!]]></url>
    <phish_id>484086</phish_id>
    <phish_detail_url>
     <![CDATA[http://www.phishtank.com/phish_detail.php?phish_id=484086]]>
    </phish_detail_url>
    <submission>
     <submission_time>2008-08-06T01:49:24+00:00</submission_time>
    </submission>
    <verification>
     <verified>yes</verified>
     <verification_time>2008-08-06T02:31:08+00:00</verification_time>
    </verification>
    <status>
     <online>yes</online>
    </status>
   </entry>
```

[그림 5-5] Phishtank XML Code 예시

```
void fileOpen(TObject *Sender)
{
        if (OpenDialog->Execute()) {
            Caption = "XmlTagExtraction - " + ExtractFileName(OpenDialog->FileName);
            m_OpenMode = OpenDialog->FilterIndex == 2 ? otfUTF16 : otfUTF8;
            StringToWideChar(OpenDialog->FileName, filename, sizeof(filename));
            HANDLE hFile = OpenReadFile(filename);
            if (otfUTF16 == m_OpenMode) {
                CWStringFileReader file_stream(hFile);
                root = parser.Parse(file_stream);
            }
            else {
                CByteFileReader file_stream(hFile);
                CUTF8DecodeStream utf8_stream(file_stream);
                root = parser.Parse(utf8_stream);
            }
            XmlTagTree->Items->BeginUpdate();
            XmlTagTree->Items->Clear();
            if (NULL != root) AddXmlNodes(NULL, root);
            XmlTagTree->Items->EndUpdate();
            CloseFile(hFile);
        }
}
void AddXmlNodes(TTreeNode* trNode, CSXmlNode* parent)
{
        TTreeNode* sub_node;
        if (parent->HasChildren()) {
            TSXmlNodes& nodes = parent->Children();
            TSXmlNodes::iterator node;
            for (node = nodes.begin(); node != nodes.end(); node++) {
                sub_node = XmlTagTree->Items->AddChild(
                        trNode, wstringToString(&(*node)->Name()[0])
                );
                sub_node->Data = static_cast<void*>(*node);
                AddXmlNodes(sub_node, *node);
            }
        }
}
void ShowNodeData(CSXmlNode& xml_node)
{
        EditXmlTagData->Lines->Clear();          // Tag Data
        EditXmlTagData->Lines->Add( wstringToString(&xml_node.Data()[0]) );
        XmlAttrList->Items->BeginUpdate();       // Attributes
        XmlAttrList->Items->Clear();
        TSXmlTagAttributes& const attrs = xml_node.Attributes();
```

```
        TSXmlTagAttributes::iterator attr =    attrs.begin();
        for (int i=0; i < attrs.size(); i++, attr++) {
                TListItem* item =    XmlAttrList->Items->Add();
                item->Caption =    wstringToString(attr->first);
                item->SubItems->Add(    wstringToString(attr->second) );
        }
        XmlAttrList->Items->EndUpdate();
}
```

[그림 5-6] XML Source Code 자동 업데이트 활용 방안

3. 시스템 개발

3.1 클라이언트 측 모듈

클라이언트 모듈을 개발하기 위한 개발 시스템 환경은 아래 표와 같다.

[표5-2] 클라이언트 모듈 개발 환경

구분	사양
개발 언어	Visual C++
시스템 사양	- CPU : Pentium 4, 2.13GHz - Memory : 2.00G - OS : Windows XP, IE 6.0

페이지 위험도 측정은 매 페이지마다 이루어지는 검사로서 피싱 사이트에서 가장 많이 쓰이는 URL 변조의 위험을 검사해주고, 사용자의 민감한 정보를 요구하는지에 대한 검사를 한다. [표 5-3]은 본 시스템의 클라이언트 측에서 점검해주는 피싱 위험 유형이다.

[표 5-3] 피싱 위험 유형 분석

분석 항목	위험요소	분석법	검출	위험 검출시
물리적	IP 주소로 이루어진 URL	URL 분석	도메인이 숫자인 경우	Y
	Forged Domain		최상위 도메인 2개이상	R
	IDN Spoofing - ASCII Character Check		사용되는 Character Code 범위 지정	Y
	특수기호 사용 URL		‘,’, ‘$’, ‘#’ 등 기호 검색	R
	비 정상적으로 긴 도메인주소		일정 크기 이상의 도메인	R
	URL Redirection Bad Domain Friendly Login URL		지정된Keyword 검색 (Keyword : 제1금융권 지정)	Y
내부적	Input Type Pattern	HTML 분석	주민번호, 카드번호, 계좌번호 입력 요구 검출	Y
	URL Address Spoofing %00, %01		%00, %01검출	R
	href link(URL) Check - IDN Spoofing : ASCII - Unicode Encoding		ASCII 범위 검색	R
	ISO-8805-1 Encoding		인코딩 정보 분석	R
외부적	Nation Check	접속 IP 분석	페이지의 IP 분석	Y

Y : 팝업 메시지 R : 피싱 경고

3.2 서버 측 모듈

서버측 구현은 아래와 같은 개발환경에서 구축된다.

[표5-4] 서버측 개발 환경

구분	사양
개발 언어	Visual C++
DBMS	MySQL 5.0
시스템 사양	- CPU : Pentium 4, 2.13GHz - Memory : 2.00G - OS : Windows XP,

클라이언트 단에서 1차적으로 접속하고 있는 웹 페이지에 대한 신뢰도 점검을 한 결과 사이트에서 위험 요소가 검출된 경우, 위험성에 따라 신호등으로 경고만 주는 경우와 좀 더 신뢰도 있는 검사 여부를 사용자에게 다시 묻는 두 가지 단계로 나누어진다.

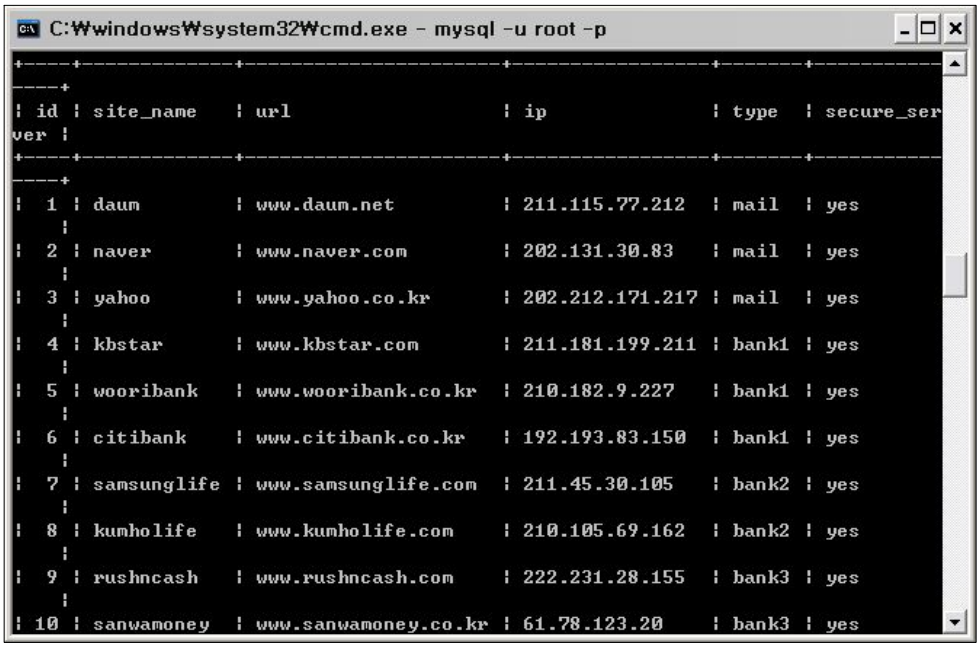
사용자가 신뢰도 있는 검사를 요청하게 되면 본 시스템의 서버로 연결된다. 여기에서는 클라이언트 측에서 추출된 위험요소에 맞추어 서버에 연결된 DB와 비교하여 사이트의 위험성을 재점검한다.

DB는 안전한 사이트들로 이루어진 Website White List(WWL)와 위험한 사이트들의 목록인 Website Black List(WBL)로 나뉜다.

3.2.1. WWL DB 설계

WWL은 웹 메일서비스를 제공하는 대표적인 사이트들과 금융권 사이트들의 2가지 부분으로 구성되어있다.

이는 피싱 사고가 E-mail을 통하여 금융권을 대상으로 이루어진다는 점과 많은 사람들이 이용하는 서비스 중 높은 신뢰도를 요구하는 서비스 2가지를 선택한 것이다. 여기서 은행권은 제1금융권 ~ 제3금융권을 표기하여 보안 솔루션 구축이 제1금융권에 비해 비교적 허술한 제3금융권은 더 주의하도록 한다.



id	site_name	url	ip	type	secure_server
1	daum	www.daum.net	211.115.77.212	mail	yes
2	naver	www.naver.com	202.131.30.83	mail	yes
3	yahoo	www.yahoo.co.kr	202.212.171.217	mail	yes
4	kbstar	www.kbstar.com	211.181.199.211	bank1	yes
5	wooribank	www.wooribank.co.kr	210.182.9.227	bank1	yes
6	citibank	www.citibank.co.kr	192.193.83.150	bank1	yes
7	samsunglife	www.samsunglife.com	211.45.30.105	bank2	yes
8	kumholife	www.kumholife.com	210.105.69.162	bank2	yes
9	rushncash	www.rushncash.com	222.231.28.155	bank3	yes
10	sanwamoney	www.sanwamoney.co.kr	61.78.123.20	bank3	yes

[그림 5-7] WWL DB 구축 화면

3.2.2 WBL DB 설계

WBL DB는 신뢰성 있는 해외 Anti Phishing 사이트에서 무료로 제공하는 피싱 사이트의 목록을 이용한다. 이는 대부분의 피싱 사이트의 진원지가 해외, 특히 미국에 많다는 점을 고려하면 많은 해외발 피싱 사이트를 막을 수 있을 것이다. WBL DB는 날마다 업데이트 되도록 하여 정보의 최신성을 항상 유지하도록 한다.

아래의 [그림 5-8]은 PhishTank에서 제공되는 피싱 사이트의 XML 문서의 일부이다. 이를 이용하여 DB에 데이터를 넣는 알고리즘은 [그림 5-9]이다.

[그림 5-8] PhishTank에서 제공되는 XML 페이지

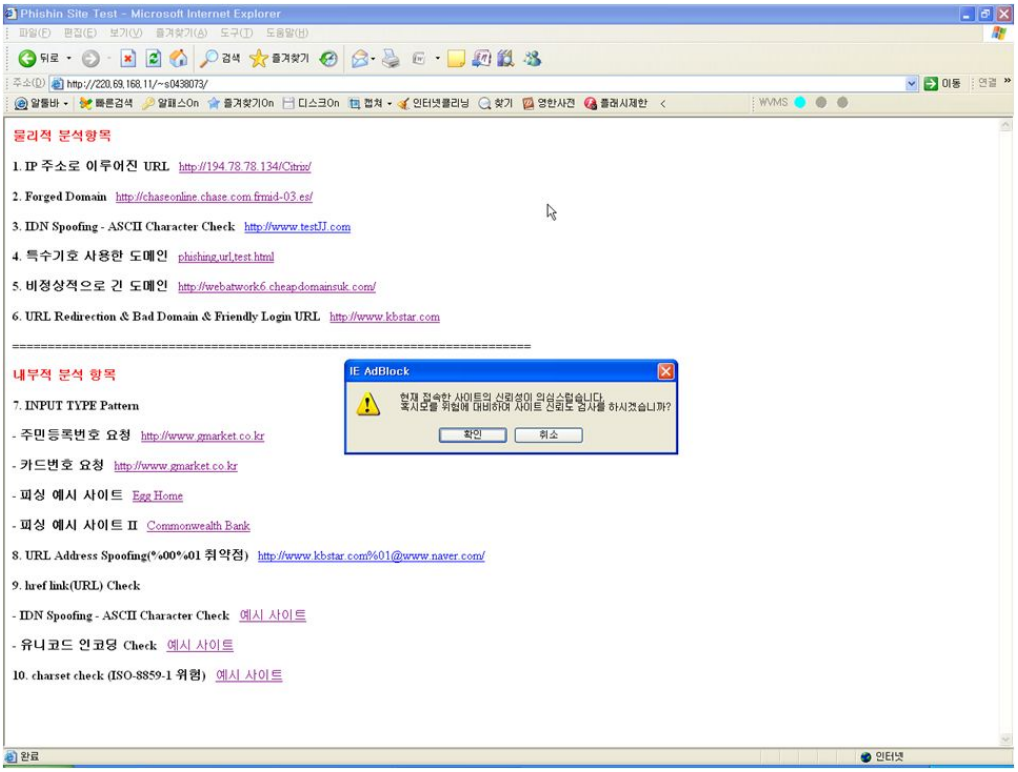
```
//entry의값중필요한항목추출
b = strtemp.AllocSysString( );
bstr = SysAllocString(b);

HRCALL(pXMLDom->selectSingleNode(bstr, &pNode), "dom->selectSingleNode: ");

        if (!pNode) {
        ReportParseError(pXMLDom, "Calling selectSingleNode");

        }
        else {

                if (bstr) SysFreeString(bstr);
                HRCALL(pNode->get_nodeName(&bstr)," get_nodeName ");
                dprintf("Node, <%S>:₩n", bstr);
                if (bstr) SysFreeString(bstr);
                HRCALL(pNode->get_xml(&bstr), "get_xml: ");
                dprintf("₩t%S₩n₩n", bstr);
        }

USES_CONVERSION;
LPCSTR cmsg = OLE2CA(bstr);
msg = (LPSTR)cmsg;

//앞에부분문자자르기http://이후값만저장
char* AnaUrl = "http://";       //고정


char* StrUrl = strstr(msg, AnaUrl);

//뒤에부분문자자르기
int StrLeng= strlen(StrUrl);
int ReStrLen = StrLeng - 9; //앞에뗀문자길이에서뒤에길이9 만큼빼고

char ReStr[300];
char *ptr;
ptr = strncpy( ReStr, StrUrl, ReStrLen);

//DB 넣기
ConnectDB.DB_Update(j,ptr);
```

[그림 5-9] XML 문서의 DB 업데이트

4. 화면 구성

1) URL 입력 후 신뢰도 검사 여부에 대한 의사 질문

웹 브라우저에 URL을 입력 후 접속한 사이트에서 위험 요소가 검출되면 사용자에게 신뢰성 검사 여부를 묻고 이에 대한 사용자의 의견에 따라 신뢰도 검사가 이루어진다.



[그림 5-10] 신뢰도 검사 여부 선택 창

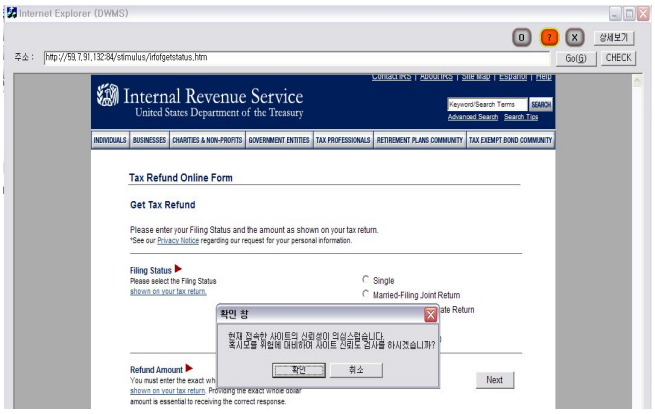
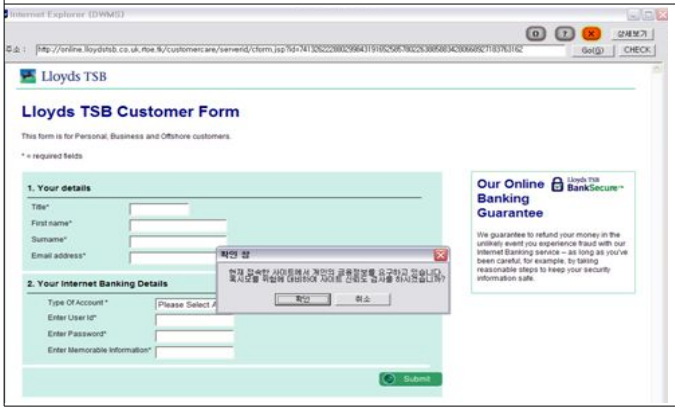
2) WWL DB에 들어있는 정상 사이트 접속 시

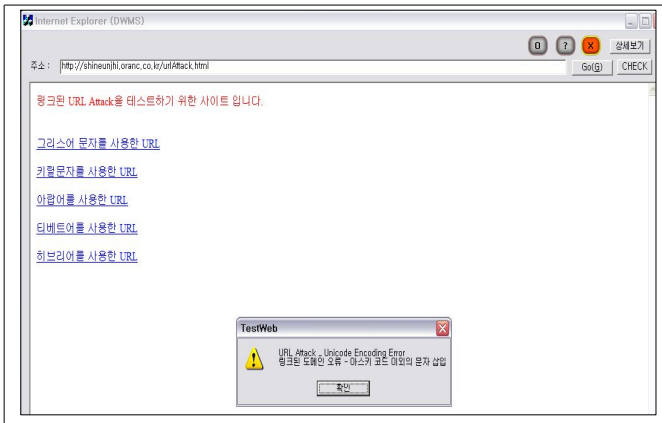
[그림 5-11] WWL DB에 있는 안전한 사이트 접속 시

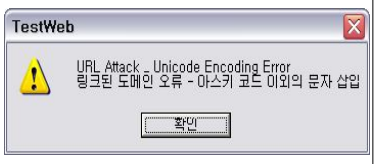
3) 의심되는 웹 사이트 신뢰도 검사 결과

신뢰도가 의심되는 사이트에서 신뢰성 검사에 동의하면, 검사 결과에 대한 내용을 아래 표와 같이 팝업 창으로 안내해준다.

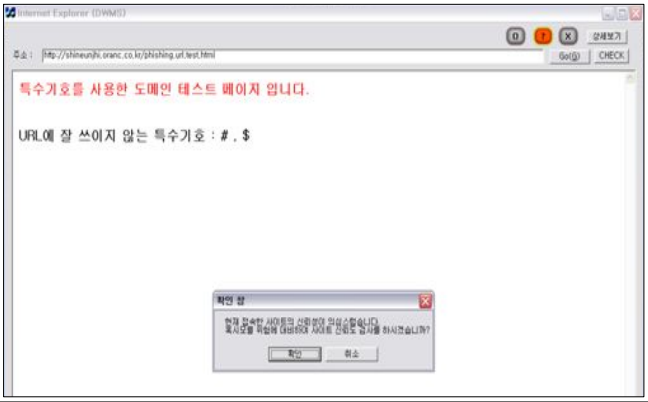
[표5-5] 의심되는 웹 사이트 신뢰도 검사 결과

1. 물리적 분석 항목 위험성 판단에 의한 위험 요소 검출	
	 ① IP 주소로 이루어진 URL
	 ② Forged Domain 대응 화면

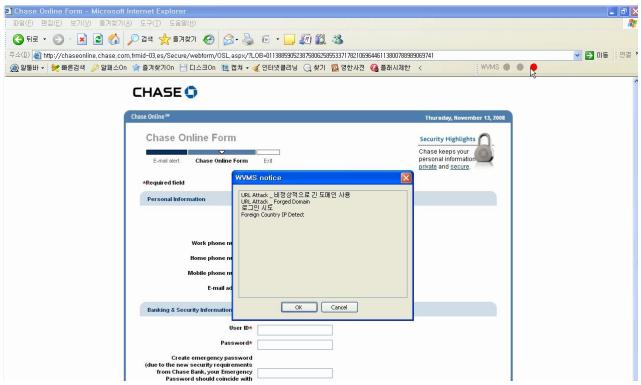




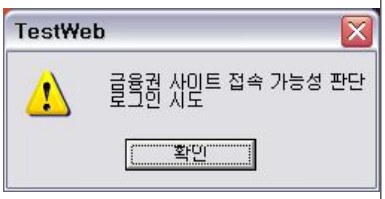
③ IDN Spoofing 대응 화면



④ 특수 기호 검출 화면

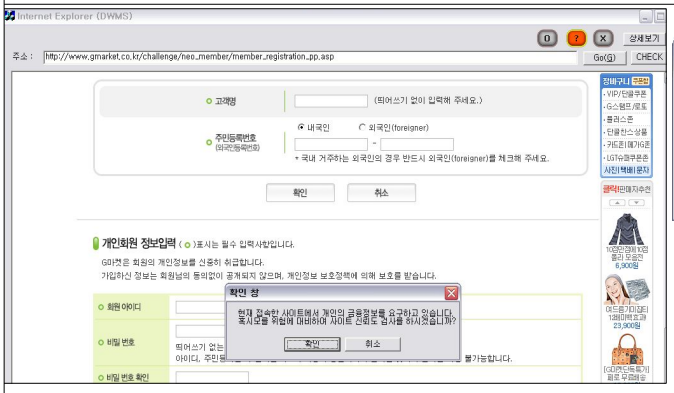


⑤ 긴 도메인 주소 검출 화면

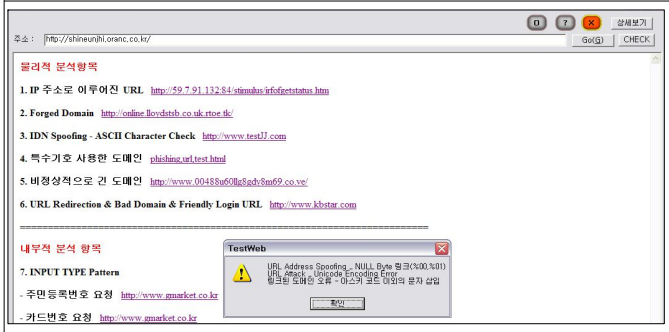


⑥ Keyword 공격 대응 화면

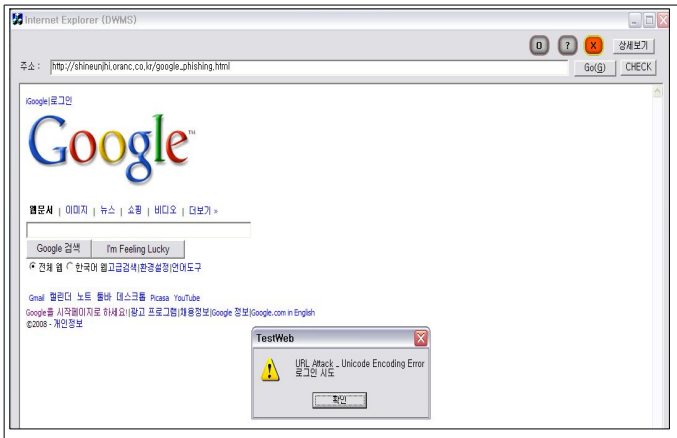
2. 내부적 분석 항목 위험성 판단에 의한 위험 요소 검출



① Input Type Pattern 분석을 통한 개인정보, 금융정보 입력요구 주의 안내



② HTML herf 링크 태그 내 URL Address Spoofing 검출 결과



③ UNICODE 인코딩 점검 결과

3. 외부적 분석 항목 위험성 판단에 의한 위험 요소 검출

① 서버가 해외에 있는 경우

Ⅶ. 결론

개인 정보 유출 시도 및 침해 사례는 날로 증가하고 있으며 현재 웹 환경 내 침해 사고 중 가장 많은 부분을 차지하고 있다. 특히 피싱은 피해자의 직접적인 금전적인 손실을 유발할 수 있다는 점에서 문제점이 크다.

이에 본 연구에서는 피싱 및 기타 개인 정보 침해 사고를 예방하기 위하여 웹사이트 보안 수준 확인 시스템 구축을 위한 연구를 진행하였다. 특히 본 시스템은 1차적으로 웹 페이지에 대한 물리적, 내부적, 외부적 분석을 실시간으로 점검해주고, 점검된 결과에 따라 사용자가 원할 시엔 2차적으로 DB 검색을 통해 좀 더 신뢰할 만한 웹 페이지의 신뢰성 결과를 제공할 수 있도록 하였다.

웹 페이지의 물리적 위험성 판단은 URL 주소 분석을 통해 이루어진다. IP주소로 이루어진 URL공격, Forged Domain공격, URL Redirection기능 이용 공격, Friendly Login URL, Bad Domain Name공격, ASCII Character공격, Spoofing E-mail공격, 특수기호공격, 비정상적으로 긴 도메인 공격 등을 막아주며 각 항목이 검색되었을 경우 사용자에게 빨간 불빛을 통한 경고나 팝업 메시지를 통해 2차적인 신뢰도 검사를 실시하도록 했다.

내부적 위험성 판단 검사에서는 개인정보 요구 구분, URL Address Spoofing, IDN Spoofing, 유니코드 인코딩을 통한 웹 페이지 내부의 Link URL을 분석하고, Input Type Pattern, Submit Pattern 분석 등을 통해 웹 페이지의 내부적 위험성을 판단해줌으로써 신뢰성 검사를 실시한다.

마지막으로 외부적 분석으로 해외에 서버를 두고 있는 사이트의 경우 사

용자에게 이를 알려줌으로써 해외사이트 접속 시에 사용자가 개인정보보호에 더 주의를 요할 수 있도록 하였다.

본 연구를 통해 제안한 웹사이트 보안수준 확인 시스템은 접속하는 각각의 URL 마다 이루어지기 때문에 기존 피싱 및 기타 개인 보안 침해 대응 기법과 비교하여 사전 탐지 및 대응이 가능 하다는 장점을 지닌다. 또한 정량적, 정성적 위험요소를 고려한 측정 방안을 수립하여 웹 사이트 보안 수준 점검을 위한 계량적 위험도 측정 알고리즘과, 신뢰성 검사 모듈을 통해 화이트 리스트 데이터베이스를 구축하고 관리하며 사용자 컴퓨터에 서버와 통신하는 사용자 모듈을 구축함으로써 개인 정보 침해 사고를 최소화할 수 있을 것으로 기대된다.

참 고 문 헌

[1] “2007국가정보보호백서”, 국가정보원·정보통신부, 2007.

[2] 김창곤, “2003 개인정보보호백서”, 한국정보보호진흥원, 2004.

[3] McAfee, "Phishing & Pharming", 2005.

[4] Phishing Attacks Escalated in 2007, Gartner Survey

[5] 최양서, “사회공학적 공격방법을 통한 개인정보 유출기술 및 대응방안 분석”, 정보보호학회지 제16권 1호, 2006.

[6] 홍승필, “유비쿼터스 컴퓨팅 보안”, 한빛미디어, 2006.

[7] 김덕진, “피싱과 안티피싱 기술의 동향”, 주간기술동향 통권 1362호, ETRI, 2008.

[8] 이영석, “웹 어플리케이션 보안 기술 동향”, 정보보호학회지 제15권 1호, pp.83-89, 2005.

[9] 정진미, “웹 애플리케이션 보안에 관한 고찰”, 한국정보과학회 가을 학술발표논문집 Vol.34, No2. pp.66-70, 2007.

[10] New Universal Man-in-the-Middle Phishing Kit Discoverd, RSA Alert, http://www.rsa.com/press_release.aspx?id=7667

[11] 이응용, “피싱(Phishing) 위협 및 대응 방안”, ITFIND 주간기술동향 통권 1237호, ETRI, 2006.

[12] Tom N. Jagatic, “Social phishing”, Communications of the ACM , Vol. 50, Issue 10, 2007.

[13] Internet violation accident trends and analysis, KrCERT/CC, 2008.

[14] http://www.antiphishing.org/reports/apwg_report_july_2007.pdf
[15] http://www.softforum.co.kr
[16] http://www.antiphishing.org
[17] http://www.phishtank.com
[18] Ye Cao, "Anti-phishing based on automated individual white-list", DIM '08: Proceedings of the 4th ACM workshop on Digital identity management, pp.51-60, 2008.
[19] Mohamad Badra, “Phishing Attacks and Solutions”, '07: Proceedings of the 3rd international conference on Mobile multimedia communications, ACM International Conference Proceeding Series Vol.329, Article No.42, 2007.
[20] Amir Herzberg, Ahmad Jbara, “Security and identification indicators for browsers against spoofing and phishing attacks”, ACM Transactions on Internet Technology, Vol. 8, No4, pp.1-45, 2008.
[21] Dieter Gollmann, "Securing Web applications", ELSEVIER Information Security Technical Report, Vol. 13, Issue 1, pp.1-9, 2008.
[22] Rechna Dhamija, "Why Phishing Works", ACM, CHI '06 : Proceedings of the SIGCHI conference on Security in computing systems, April 22-27 2006, pp.581-590, 2006.
[23] 성재모, “최근 주요 해킹 피해 동향과 대응 방안”, 정보보호학회지 제6권 1호, pp.80-84, 2006.