Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.23, NO.6, Dec. 2013 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2013.23.6.1121

악성코드 유포 네트워크 분석을 위한 멀티레벨 에뮬레이션

최 상 용,[1†] 강 익 선,[1] 김 대 혁,[1] 노 봉 남,[2] 김 용 민[2‡]
[1]한국과학기술원, [2]전남대학교

Multi-Level Emulation for Malware Distribution Networks Analysis

Sang-Yong Choi,[1†] Ik-Seon Kang,[1] Dae-Hyeok Kim,[1] Bong-Nam Noh,[2] Yong-Min Kim[2‡]
[1]Korea Advanced Institute of Science and Technology, [2]Chonnam National University

요 약

최근 악성코드 유포는 단순한 개인적 피해를 넘어 3·20 사이버테러와 같은 사회적인 심각한 문제점을 야기하고 있다. 특히 취약한 웹을 통한 유포방법인 드라이브 바이 다운로드(Drive-by download) 공격은 가장 심각한 위협이 되고 있다. 따라서 드라이브 바이 다운로드 공격에 사용되는 악성코드 유포 네트워크(Malware Distribution Network, MDN)를 효과적으로 분석하는 것은 악성코드로 인한 피해를 예방하기 위해 매우 중요하다. 악성코드 유포 네트워크를 효과적으로 분석하기 위해서는 웹페이지 내에 포함된 난독화하고 은닉화한 스크립트를 식별해야 하며, 본 논문에서는 이를 식별하기 위해 멀티레벨 에뮬레이션 기법을 제안한다. 이를 통해 다양한 형태로 숨겨져 있는 유포지로 연결하는 링크를 분석하여 악성코드 유포 네트워크 분석의 기반을 제공하고자 한다.

ABSTRACT

Recent malware distribution causes severe and nation-wide problems such as 3·20 cyber attack in Korea. In particular, Drive-by download attack, which is one of attack types to distribute malware through the web, becomes the most prevalent and serious threat. To prevent Drive-by download attacks, it is necessary to analyze MDN(Malware Distribution Networks) of Drive-by download attacks. Effective analysis of MDN requires a detection of obfuscated and/or encapsulated JavaScript in a web page. In this paper, we propose the scheme called Multi-level emulation to analyze the process of malware distribution. The proposed scheme analyzes web links used for malware distribution to support the efficient analysis of MDN.

Keywords: Drive-by download, Web-based malware, Multi-level emulation, Malware Distribution Network

I. 서 론

웹 기술의 끊임없는 발전과 함께 다양한 영역에서 웹 기반 서비스들이 출시됨으로써 웹은 수많은 사람들의 일상 활동이 일어나는 공간이 되었다. 이와 더불어 사용자 활동의 기반이 되는 웹 브라우저와 관련 플러그인 어플리케이션들은 웹에서 가장 많이 사용 하는 클라이언트 소프트웨어가 되었다. 이러한 컴퓨팅 환경의 변화에 따라 악의적 공격자의 공격 형태도 달라지고 있는데, 최근에는 악성 코드를 쉽고 빠르게 퍼뜨릴 수 있는 웹 클라이언트의 취약점을 이용한 웹 기반의 악성 코드 유포 방법으로 드라이브 바이 다운로드(Drive-by download) 공격이 가장 널리 사용되고 있다[1-4].

유럽네트워크정보보호원(European Network and Information Security Agency)의 사이버 공격 동

접수일(2013년 9월 16일), 수정일(2013년 10월 10일), 게재확정일(2013년 11월 13일)
† 주저자, csyong95@kaist.ac.kr
‡ 교신저자, ymkim@chonnam.ac.kr(Corresponding author)

향 보고서에 따르면, 드라이브 바이 다운로드 공격이 최근 웹을 통한 공격 방법 중에 가장 높은 비율을 차지하고 있다[2]. 실제로 최근에는 미 NBC 방송국 홈페이지와 미 노동부 홈페이지 등 정부 기관이나 유명 웹 사이트가 해킹 되어 드라이브 바이 다운로드 공격에 의해 방문자들이 악성코드에 대규모로 감염 된 사례가 있다[3,4]. 국내에서도 2011년 SK커뮤니케이션즈 개인정보 유출사고[5], 2013년 3·20 사이버테러[6], 6·25 사이버테러[7] 등에 사용된 악성코드도 드라이브 바이 다운로드 공격으로 최초 유포된 것이 확인되었다.

공격자가 공격에 사용하는 악성코드 유포사이트로 자동으로 연결되는 웹페이지의 링크를 악성코드 유포 네트워크(MDN, Malware Distribution Network)이라 한다. 악성코드 유포 네트워크를 효과적으로 탐지하기 위해 많은 연구가 이루어지고 있으나 최근 공격자의 우회를 위한 지능적 기술에 대응하기에는 일부 한계가 있다.

본 논문에서는 기존 연구된 탐지를 위한 방법의 한계점을 해결하기 위한 분석방법인 멀티레벨 에뮬레이션(MULEM, Multi-Level Emulation)을 제안한다.

II. 관련연구

본 장에서는 제안하는 멀티레벨 에뮬레이션의 소개에 앞서 드라이브 바이 다운로드 공격을 통한 악성 코드 유포가 어떻게 이뤄지는지 알아보고 기존의 탐지방법 및 공격자의 스크립트 탐지우회 기법에 따른 기존 탐지방법의 한계점을 설명한다.

2.1 드라이브 바이 다운로드 공격

드라이브 바이 다운로드 공격은 궁극적으로 사용자 컴퓨터의 취약점을 이용하여 악성코드를 유포하는 방법이다. 공격자는 사용자 PC를 악성코드 유포사이트로 유도하기 위해 취약점을 가진 웹 서버에 SQL인젝션(SQL injection)과 같은 방법으로 침투한 후 웹 페이지에 직접 악성코드유포지로 연결 되는 코드를 삽입하거나, 광고 배너나 제3의 위젯과 연결되어 있는 링크를 변조하여 공격을 실행한다[1]. 드라이브 바이 다운로드 공격은 Fig.1.과 같이 사용자의 컴퓨터가 해킹된 웹 사이트인 최초 접속지에 접속하면 공격자가 삽입해 놓은 유도코드 또는 변조한 링크에 의해 수단계의 경유지를 거쳐 자동으로 악성코드 유포사이트로

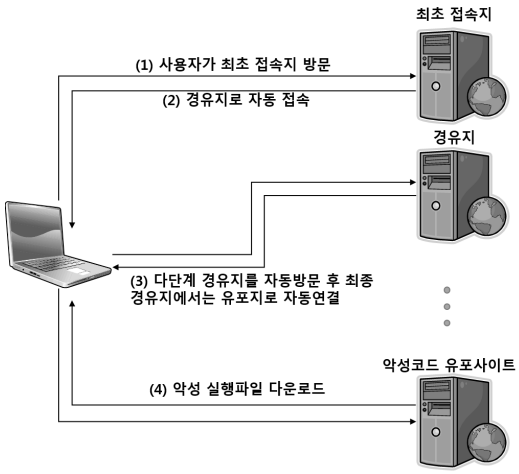

Fig.1. Malware Distribution Network(MDN)

연결된다. 이를 위해 공격자는 웹페이지 내 iframe, frame 태그(tag)를 사용하거나 HTML 메타 태그의 http-equiv 파라미터의 값을 "refresh"로 설정하고 content 파라미터의 값에 목적지 식별주소(url)을 추가하는 방법 또는 자바스크립트 document.write(), document.writeln() 과 같은 함수를 사용한다.

사용자 컴퓨터가 유포사이트로 접속되면, 사용자 컴퓨터의 웹 브라우저나 어도비 플래시 플러그인과 같은 어플리케이션 취약점이 존재하는 사용자 컴퓨터로 실행 파일이 다운로드 되고 컴퓨터는 악성코드에 감염된다.

2.2 드라이브 바이 다운로드 공격 탐지 기법

드라이브 바이 다운로드 공격을 탐지하기 위해 기존 연구는 악성코드 유포지를 분석하는 것을 위주로 하고 있다. 기존 연구는 정적분석과 동적 분석 방법으로 나누어진다. 정적분석은 웹 페이지를 렌더링하지 않고 패턴 매칭과 웹 페이지의 메타 정보를 분석하여 웹 페이지의 악성 여부를 분석하는 기법이며 동적 분석은 웹 페이지를 실제로 렌더링 하여 악성행위를 분석하는 방법이다. 동적 분석은 크게 웹 페이지와 자바스크립트 에뮬레이션 환경을 통해 웹 페이지를 방문하여 악성 코드가 다운로드 될 수 있는 시그니처가 발견되면 탐지하는 Low-interaction 방식과 가상 머신을 사용하여 웹 페이지를 직접 방문하고 파일시스템, 레지스트리, 프로세스, 네트워크 상태 등의 변화를 측정, 악성 코드의 실행 여부를 탐지하는 High-inter-

Table 1. The methods for MDN analysis

기법 분류 / 시스템 명	정적 분석		동적 분석		
			Low-interaction		High-interaction
	패턴 매칭	메타 정보 분석	DOM 파싱	자바스크립트 에뮬레이션	가상머신 기반
URL learning[8,9]	X	O	X	X	X
SafeBrowsing[10]	X	O	X	X	O
JSAND[11]	O	X	O	O	X
WebPatrol[12]	O	X	O	X	X
SpyProxy[13]	O	X	X	X	O
Landing Page[14]	O	X	O	X	O

action 방식으로 나눠지고, 세부적으로는 돔(DOM, Document Object Model) 파싱, 자바스크립트 에뮬레이션, 가상머신 기반 분석으로 분류된다. Table.1.과 같이 알려진 대부분의 시스템들은 정적과 동적 분석 기법을 함께 사용한다.

Ma et al.[8,9]은 URL과 DNS, WHOIS 등 호스트 관련 메타 정보에서 특징을 추출하여 패턴 매칭을 하거나 웹 페이지와 연결된 링크, 내용을 구성하는 태그와 자바스크립트 코드, 호스트 정보 등에서 특징을 추출하여 기계 학습 기법으로 학습시킨 후 새롭게 수집한 웹 페이지의 악성 여부를 판단한다.

Safebrowsing[10]은 웹 페이지에 대한 국가별 호스팅 정보, URL을 방문했을 때의 감염경로, 바이러스 백신의 탐지결과, 서버 스크립트의 버전 등을 통계 분석한 후 가상머신을 이용하여 검증한다.

JSAND[11]는 악성 스크립트를 포함하는 웹 페이지의 특징을 추출하여 기계 학습과 변칙 탐지 기법을 에뮬레이션 함께 활용하여 공격을 수행하는 웹 페이지 및 스크립트를 탐지한다. JSAND는 자바스크립트와 PDF를 대상으로 해당 주어진 웹 페이지의 악성 여부를 판단한다. JSAND의 경우 주어진 검사 페이지 외에 에뮬레이션을 통해 검사 페이지에 포함된 링크정보를 같이 분석한다.

Web Patrol[12]은 웹 기반의 악성 코드에 감염되는 시나리오를 트리 형태의 그래프로 정의하고, 탐지 되는 시나리오를 수집, 캐싱 하여 추후 분석 과정에서 재현 할 수 있는 시스템을 제안 하였다.

SpyProxy[13]는 가상머신 기반의 웹 페이지 탐지를 할 수 있는 웹 프록시 구조를 제안 하였다. 클라이언트로부터 발생하는 HTTP 요청을 웹 프록시 서버가 간단히 정적 분석을 한다. 그 결과가 안전하지 않다고 판단되면 프록시의 가상 머신에서 동적 분석을 하고 클라이언트로의 응답 페이지를 결정한다.

Wang et al.[14]은 악성코드 유포 네트워크에서 최초 방문지를 탐지 하는 방법을 제안 하였다. 이미 탐지 된 최초 방문지의 특징을 추출하여 분류기와 클러스터링 모듈로 학습시키고, 새로운 페이지가 수집되었을 때, 기존의 악성 코드 유포 네트워크의 일부가 될 수 있는지 판단한다.

2.3 한계점 및 개선방안

공격자는 공격에 사용 되는 악성코드 유포지로 유도하는 유도코드가 쉽게 탐지 되지 않도록 숨긴다. Fig.2.와 같이 document.write() 함수의 파라미터로 iframe과 같은 태그를 사용하는 방법이나, 함수를 사용하지 않고 웹 페이지 내에 iframe 태그를 삽입하고 태그가 포함된 코드 자체를 난독화 하는 것과 같은 단순 방식은 분석 및 탐지하기가 용이하다.

탐지를 우회하기 위해 공격자가 사용하는 코드 우회의 방법 중 하나는 Fig.3.과 같이 웹 페이지 또는 자바스크립트 내에 iframe과 같은 태그를 document.write() 함수의 파라미터로 삽입한 다음 그 자체를 난독화 하는, 보다 복잡한 방법을 사용한다. 또는 이미지 파일 내에 링크를 삽입하는 것과 같이 비정상 파일을 사용한다. Fig.4.는 이미지 파일이지만 실제 악성코드유포지로 향하는 링크가 포함된 파일의 예시이다.

```
if (document.cookie.indexOf('hsblm=') == -1) {
    var expires = new Date();
    expires.setTime(expires.getTime() + 12 * 60
* 60 * 1000);
    document.cookie =
'hsblm=Yes;path=/;expires=' +
expires.toGMTString();
    document.write("<iframe
src=http://www.hwanni.co.kr/shop/data/good
s/pop/ye.html width=0 height=0></iframe>")
}
```

Fig.2. Automatically connect to other web page using javascript function

```
eval(function(p,a,c,k,e,d){e=function(c){return
c.toString(36)};if(!''.replace(/^/,String)){while(c--)
{d[c.toString(a)]=k[c]||c.toString(a)}k=[function(e)
{return d[e]}];e=function(){return'WWw+'};c=1};while(c--
){if(k[c]){p=p.replace(new
RegExp('WWb'+e(c)+'WWb','g'),k[c])}}return p}
('d(3.7,c(W'4W')==-1){e 2=f
h();2.g(2.i()+b*5*5*8);3.7=W'4=9;a=/;2=W'+2.j();3.p(s("
<6 r=u://w.v.t.q/l/k.m n=0 o=0>
</6>"))}',33,33,'||expires|document|rinixx|60|iframe|co
okie|1000|Yes|path|24|indexOf|if|var|new|setTime|Date|g
etTime|toGMTString|uk|club|asp|width|height|write|kr|sr
c|unescape|co|http|diyhard|img'.split('|'),0,{}))
```

Fig.3. Obfuscated script

공격유도 코드를 숨기는 보다 지능적인 방법은 유포지로 연결되는 공격코드를 윈도우 객체 이벤트 속성으로 숨기는 동적코드이다. Table.2.는 지능적 기법의 대표적인 예이다. 공격자는 “fr1”과 “fr2” 두 개의 프레임을 만든다. “fr2”프레임은 가로와 세로를 “0”으로 주어 숨김 프레임으로 만든다. Table.2.에서 “fr2” 프레임의 src속성은 “NULL”이다. 다음 라인에서 “fr1”이 브라우저에 읽혀질 때 온로드(onload) 이벤트로 인해 “fr2” 프레임의 src가 “success.html”로 바뀐다. 이와 같은 형태는 document.write()함수와 같이 HTML문서의 돔 구조를 동적으로 갱신 하는 돔 쓰기 행위이다.

악성코드 유포지는 공격자가 제작해 놓은 웹 사이트이며, 공격자는 지속적인 유포를 위해 유포지 정보를 정기적으로 변경한다. 따라서 악성코드로 인한 감염을 예방하기 위해서는 유포지뿐만 아니라 유포지로 자동으로 연결되는 최초접속지 및 경유지 등 악성코드 유포 네트워크 자체를 탐지하여 접속을 차단하는 것이 중요하며, 이를 위해서는 먼저 탐지우회 기법이 적용된 지능화된 자동연결 태그 및 자바스크립트를 정확하게 분석할 수 있어야 한다.

하지만 기존의 연구 결과는 주어진 페이지가 악성 페이지인지를 확인하는데 중점을 두기 때문에 다운로드 행위가 발생되지 않는 최초접속지 및 경유지가 입력되었을 경우 악성코드 유포 네트워크의 부분으로 분석하지 않는다. 이는 유포지로 연결되는 전체 네트워크를 분석할 수 없는 한계점이 있다. 또한, Table.2.와 같은 지능적 기법이 적용된 동적코드 분석하는데 한계가 있다(분석에 대한 세부결과는 시험결과 참조).

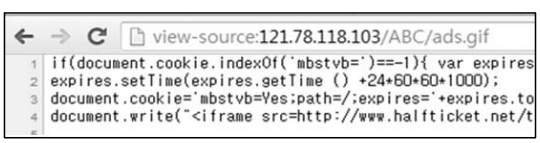

Fig.4. Image file embedded malicious links

Table 2. Example of dynamic code-behind

```
〈HTML〉
〈HEAD〉
〈TITLE〉onload test〈/TITLE〉
〈/HEAD〉
〈BODY〉
    〈iframe id="fr2" border=0 height=0 width=0〉〈/i
  frame〉
    〈iframe id="fr1" src="http://ikseon.kr" onload=
  "fr2.location.href='success.html' "〉〈/iframe〉
〈/BODY〉
〈/HTML〉
```

이와 같이 지능적 기법이 적용된 태그와 스크립트에 대해 기존 연구 중 가상머신을 사용하는 분석방법은 실제 웹 페이지를 실행하고 시스템 상에 나타나는 파일, 레지스트법을 사용하기 때문에 탐지할 수 있는 가능성이 있다. 하지만 악성행위는 실제 취약점 존재하는 컴퓨터에서 일어나므로 다양한 취약환경의 가상머신을 구축하여 검증하여야 한다.

이와 같은 한계점을 해결하기 위해서는 돔 갱신과 같은 방법을 사용하여 수 개의 웹 사이트로 연결된 악성코드 유포 네트워크의 링크정보를 반복적으로 추적하고, 다양한 형태로 적용된 회피방법을 분석하여야 한다.

본 논문에서 제안하는 멀티레벨 에뮬레이션은 악성코드 유포 네트워크의 구성요소 중 최초접속지와 경유지 정보까지를 분석하는데 목적이 있으며, 콘텐츠 기반 정적분석과 low-interaction 방법인 스크립트 에뮬레이션을 사용하여 웹 사이트사이 비정상 링크의 반복추적을 가능하게 하고 다양한 형태로 숨겨져 있는 악성코드유포지로 유도시키는 링크를 분석하여 악성코드 유포 네트워크 분석의 기반을 제공한다.

III. 멀티레벨 에뮬레이션

이번 장에서는 본 논문에서 제안하는 웹 페이지 순회 구조인 멀티레벨 에뮬레이션 기법을 소개한다. 제안하는 기법은 스크립트를 분석하기 위한 에뮬레이터와 복합기법이 적용된 웹 페이지 내 스크립트 및 태그를 추적하기 위한 순환구조로써 웹 페이지 내 포함된 스크립트를 반복검사하여 최초접속지로부터 악성코드 유포지에 이르는 전체 경로를 추적한다.

3.1 멀티레벨 에뮬레이션 구조

Fig.5.의 플로차트는 멀티레벨 에뮬레이션 동작의 기본적인 과정을 나타낸다. 검사할 대상이 존재하면

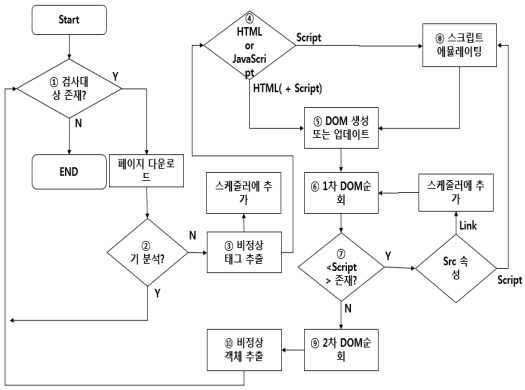

Fig.5. The procedure of MULEM

해당 웹 페이지를 다운로드 하고 그 페이지가 이미 분석 되었는지 확인한다. 이 과정을 통해 불필요한 탐색 과정을 축소한다. 이전에 분석하지 않은 페이지는 웹 페이지 내에 숨김 iframe과 같은 비정상 태그의 존재 여부를 확인하여 데이터베이스에 저장한다. 비정상 태그의 판단 기준은 Table 3.과 같다. 이후 최초 접속지의 웹 페이지는 HTML 페이지라고 판단하고 분석되는 웹 페이지는 해당 페이지로 향하는 링크가 어떤 태그로부터 추출 되었는지를 통해 페이지 유형을 판단하게 된다. 유형에 따라 에뮬레이션이 어떻게 진행될지 결정한다. 만약 페이지가 HTML 코드로만 혹은 자바스크립트와 혼재되어 있을 경우에는 돔 트리를 생성한다. 돔 트리는 웹 페이지 내에 포함된 모든 태그의 이름과 속성을 표시해 주는 구조이며, 1차 돔 순회는 생성한 돔 트리를 순회하며 Table 3.에 명시된 것과 같이 속성이 숨김인 iframe 과 같은 비정상 태그와 redirection, script 태그 등을 추출한다. script 태그를 발견하면 src 속성의 유무를 확인하고 src 속성이 존재 한다면 연결된 자바 스크립트를 다운로드 할 수 있도록 스케줄링 한다. 만약 src 속성 없이 내장코드가 존재 한다면 스크립트 에뮬레이션 과정을 진행한다.

스크립트 에뮬레이팅 모듈은 오픈 소스 자바스크립트 엔진인 SpiderMonkey[15]를 기반으로 구현하였다. 스크립트 에뮬레이터는 엔진 초기화→스크립트 로딩→엔진 실행→결과 분석 및 반환의 과정을 수행한다. 엔진 초기화 과정에서는 인터넷 익스플로러 모듈, 돔 관리 모듈, 쿠키(Cookie) 모듈 등을 초기화 한다. 또한 자바스크립트에 의해 실행 될 함수를 등록한다. 초기화 과정이 끝나면 입력한 스크립트를 로드 하고

Table 3. Critical value for detection of anomaly TAG

구분	단위(속성)	속성값
스타일 및 태그속성의 넓이와 높이	em	0.5 보다 작음
	ex	1 보다 작음
	px	2 보다 작음
	%	1 보다 작음
	pt	2 보다 작음
스타일 속성	display	none
	visibility	hidden

엔진에 넘긴다. 엔진은 입력한 스크립트를 에뮬레이션 하고 그 결과 돔 객체 내에서 쓰기 행위가 감지되면 돔 객체를 갱신 한다. 갱신한 돔 트리를 이전과 같은 방식으로 순회를 하게 되고, 이 과정은 스크립트 태그가 더 이상 발견 되지 않을 때 까지 반복된다(1차 돔 순회).

스크립트 태그가 존재하지 않을 경우 2차 돔 순회를 수행하고, 그 결과를 비정상 객체 탐지 모듈로 넘겨준다. 비정상 객체 탐지 과정에서는 돔 트리 순회 과정에서 발견한 object나 applet 태그를 검사한다. Table 3.과 같이 스타일 및 태그 속성이 숨김이거나 매우 작게 설정된 비정상적으로 삽입된 객체의 존재 여부를 판단하여 데이터베이스에 저장한다. 비정상 객체 탐지를 마치면 한 페이지에 대한 검사가 끝나게 되고 더 검사할 대상이 있는지 여부에 따라 추후 진행 과정이 결정된다.

멀티레벨 에뮬레이션의 주요 기능을 간단히 요약하면 첫째, 웹 페이지에 포함된 다양한 콘텐츠 중 타 사이트로 연결이 가능한 객체를 찾아내어 분석한다. 둘째, 분석대상 페이지(파일)의 내용을 기반으로 분석한다. 셋 째, 돔 트리를 지속적으로 갱신하고 갱신한 돔 트리 내 포함된 비정상 링크 확인을 위해 중복검사를 수행한다. 이를 바탕으로 본 논문에서 제안하는 멀티레벨 에뮬레이션 기법과 기존 연구와의 차이점은 Table 4.와 같다.

3.2 멀티레벨 에뮬레이션 예

웹 페이지의 주소를 입력하면 멀티레벨 에뮬레이션에서는 먼저 페이지에 대한 돔 트리를 Fig.6.과 같이 생성한다. 돔 트리는 웹 페이지의 구조를 태그와 속성 형태로 나타내 준다. 돔 트리를 파싱한 결과 스크립트가 포함되어 있는 것이 확인되며, 난독화 되어 있는 스크립트가 포함되어 있다. 난독화된 스크립트는 에뮬레이터로 전송되고 에뮬레이팅 결과 document.

Table 4. Difference of the existing method and multi-level emulation

기존 연구	분석방법	주요 특징(장단점)
URL learning[8,9]	메타정보분석	초기 트레이닝 셋 필요, 렌더링 이후 실행되는 공격 분석 한계
SafeBrowsing[10]	메타정보분석, 가상머신 기반 검증	유포지 탐지가 목적이며 최종 검증에 가상머신 사용, 다양한 취약점에 취약한 환경 필요
JSAND[11]	패턴매칭, DOM구조 에뮬레이션	유포지 탐지가 목적이며, 10가지의 특성인자와 브라우저 에뮬레이터 활용
WebPatrol[12]	패턴매칭, DOM구조 분석	악성코드 유포 네트워크 수집 및 분석을 위한 재생(행위)이 주목적, 은닉된 태그 분석 한계
SpyProxy[13]	패턴매칭, 가상머신 기반 검증	웹 프락시를 이용하여 사용자와 웹서버간의 통신을 통제, 브라우저 환경차이에 의해 분석되지 않을 수 있음.
Landing Page[14]	패턴매칭, DOM구조 분석, 가상머신	초기 페이지 탐지가 목적, 초기 통계분석 후 의심되는 URL에 대해 가상머신에서 재검증
MULEM(제안기법)	패턴매칭, DOM구조, 에뮬레이션	유포 네트워크 전체 분석(초기페이지, 경유지, 유포페이지) 페이지 콘텐츠 분석 및 에뮬레이션을 사용하여 렌더링 이후 변경되는 웹 페이지까지 추적가능, 최종 악성코드 자동다운로드 한계

write() 함수의 파라미터로 iframe이 포함된 것이 확인된다.

멀티레벨 에뮬레이션은 iframe을 추적하기 위해 돔 트리를 갱신 한다. 돔 트리 갱신은 기존 "tag_ptr: 147343912"인 스크립트를 삭제하고 Fig.7.과 같이 그 자리에 iframe 에 대한 돔 구조를 삽입하는 형태로 이루어진다.

다음으로 돔 순회를 통해 iframe의 src 속성에 해당하는 웹 페이지를 다운로드하고 돔 트리를 갱신하는 과정이 스크립트가 더 이상 존재하지 않을 때까지 지속적으로 반복된다.

```
analysis url : http://samples.ikseon.kr/
---------- before start ----------
{
        "tag_ptr":      147343752,
        "tag_name":     "body",
        "tag_attr":     []
}
{
        "tag_ptr":      147343912,
        "tag_name":     "script",
        "tag_attr":     []
}
---------- before end ----------
======= function(p,a,c,k,e,d)
[function(p,a,c,k,e,d){e=function(c){return
c.toString(36)};if(!''.replace(/^/,String)){while(c--){d..]=======
- http://samples.ikseon.kr/
------------------ JS_mlms_print!! ------------------
<iframe
src=http://www.hwanni.co.kr/shop/data/goods/pop/ye.html
width=0 height=0> </iframe>
-----------------------------------------------------
======= iframe [iframe
src=http://www.hwanni.co.kr/shop/data/goods/pop/ye.html
width=0 height=0> </iframe>]=======
```

Fig.6. DOM traversal and obfuscated script

IV. 실험 및 분석

4.1 실험 방법

실험은 멀티레벨 에뮬레이션의 분석성능과 분석속도 두 가지 측면에서 실시하였다. 분석성능을 실험하기 위해 웹 페이지사이 자동이동이 가능한 6가지 방법을 정의하고 이에 대해 정확한 분석이 가능한지 여부를 제안한 기법과 유사하게 패턴매칭과 에뮬레이션을 방법을 사용하는 JSAND와 비교검증 하였고, 분석속도는 비정상 링크 및 정상링크가 혼합된 실제 환경에서 1,040개의 사이트에 포함된 70,139개의 웹 페이지에 대해 멀티레벨 에뮬레이션의 분석 속도를 측정하였다.

4.2 실험 결과

분석성능 실험결과는 Table 5.와 같다. 6가지 실험조건에 대해 제안하는 멀티레벨 에뮬레이션은 모두 분석이 가능하지만 JSAND는 동적태그에 대해 정확한 분석에 한계가 있다. Fig.8.과 같이 동적코드가 포함된 페이지의 소스를 보면 페이지 내 프레임이 3개가 존재하고, 3개의 프레임에는 각각 다른 페이지가 링크

```
- http://samples.ikseon.kr/
---------- after start ----------
{
        "tag_ptr":      147343752,
        "tag_name":     "body",
        "tag_attr":     []
}
.....
{
        "tag_ptr":      139762744,
        "tag_name":     "iframe",
        "tag_attr":     [{
                        "name": "src",
                        "value":        "http://www.hwanni.co.kr/shop/data/goods/pop/ye.html"
                }, {
                        "name": "width",
                        "value":        "0"
```

Fig.7. Update the DOM tree

Table 5. Results of the analysis of MULEM and JSAND

No	웹 페이지 연결방법	MULEM	JSAND
1	단순 태그	분석가능	분석가능
2	자바스크립트	분석가능	분석가능
3	난독화	분석가능	분석가능
4	자바스크립트 함수	분석가능	분석가능
5	비정상 파일	분석가능	분석가능
6	동적 태그	분석가능	분석불가

된다. 3개의 프레임은 온로드 이벤트에 의해 프레임이 읽혀 질 때 동일 페이지 내 다른 프레임의 src 속성을 변경한다.

JSAND는 Fig.9.와 같이 "fr2"프레임에 해당하는 "success-frame1.html"을 추출하지 못하고, "fr1"에 대해서는 실제 "success-frame2.html" 페이지가 링크되지만 이를 분석하지 못한다. 제안하는 멀티레벨 에뮬레이션에서는 Fig.10.과 같이 3개의 링크를 추출하고 분석을 수행한다. 실험결과 멀티레벨 에뮬레이션은 웹 페이지에 포함된 자동으로 이동이 가능한 방법을 JSAND에 비해 보다 세부적으로 분석할 수 있다는 것이 검증되었다.

두 번째로 일반적인 태그와 비정상 태그, 스크립트 등이 혼재되어 있는 실제 1,040개 웹사이트 70,139개 웹페이지에 대한 분석속도를 측정한 결과 Table 6.과 같이 페이지당 분석시간은 1.29초, 사이트당 분석시간은 87.5초로 확인되어 상당히 빠른 속도로 분석하고 있다. 시험을 위해 사용된 웹 페이지의 연결구조는 iframe으로 연결된 페이지 6,228개, 자바스크립트로 연결된 페이지 29,351개, 이외 A href 등 기타 링크형태 34,560개 등이 혼합되어 있는 형태이며, 멀티레벨 에뮬레이션은 이와 같은 링크에 대해 정확히 분석하고 있다. 1,040개 웹 사이트 중 난독화된 페이지로의 연결이 포함된 사이트는 701개 사이트이며 총

```
<BODY>
    <iframe id="fr2"></iframe>
    <iframe id="fr1" src="http://ikseon.kr"
onload="fr2.location.href='success-frame1.html' "></iframe>
    <iframe id="fr3" src="http://ikseon.kr"
onload="fr1.location.href='success-frame2.html' "></iframe>
</BODY>
```

Fig.8. Test page of dynamic code

Requests

URL	Status	Content Type
http://jsand.ikseon.kr/	200	text/html
→ about:blank	200	text/html
→ http://ikseon.kr/	200	text/html

Fig.9. Result of JSAND

```
analysis url : http://jsand.ikseon.kr/
mlms_href[target url] : success-frame1.html
mlms_href[target url] : success-frame2.html
analysis url : http://ikseon.kr/
analysis url : http://jsand.ikseon.kr/success-frame2.html
analysis url : http://jsand.ikseon.kr/success-frame1.html
분석 페이지 개수 : 4
페이지 분석  평균 소요시간 : 0.025955 [0.103821]
```

Fig.10. Result of MULEM

분석대상 70,139개 중 난독화 스크립트가 포함된 페이지는 15,015개로 확인된다.

추가적으로 분석대상 웹 페이지 중 실제 악성코드를 유포하는 페이지는 12개로 확인되었으며 이는 추출된 링크정보를 분석하는 과정에서 부가적으로 확인된 정보로 멀티레벨 에뮬레이션의 분석결과를 활용하여 악성코드 유포 네트워크를 탐지하기 위한 기초 자료로 활용 가치가 있음을 보여준다. 분석속도 측정결과에서 특이할 점은 악성코드 유포 페이지의 경우 분석시간이 14.8초로 일반 페이지 분석 시간 1.29초 보다 10배 이상 오래 걸리고 있다. 이는 유포페이지의 경우 사용자 컴퓨터에 존재하는 취약점을 악용하는 스크립트가 다수 포함되어 있어 이에 대한 분석시간이 상대적으로 오래 걸리는 것으로 분석된다.

Table 6. Analysis time of MULEM

구분	수(개)	분석시간(초)
전체 웹 사이트	1,040	87.50
전체 웹 페이지	70,139	1.29
난독화 포함 사이트	701	24.20
난독화 포함 페이지	15,015	1.13
악성코드 유포 페이지	12	14.80

V. 결 론

본 논문에서는 복잡한 탐지우회 기법이 적용된 웹 페이지 내의 링크를 분석하여 드라이브 바이 다운로드 공격에 사용되는 악성코드 유포 네트워크를 효과적으로 분석할 수 있는 기반이 되는 멀티레벨 에뮬레이션 방법을 제안하였다. 실험 결과 제안기법은 페이지 내에 포함된 비정상적인 형태의 링크를 반복적인 검사를 통해 효과적으로 추적하였으며, 특히 탐지를 우회하기 위해 사용되는 지능적 회피기법이 적용된 스크립트에 대해 기존 제안된 시스템 대비 효과적으로 분석할 수 있음을 보여주었다.

이를 기반으로 웹 페이지들의 링크인 악성코드 유포 네트워크를 상세하게 작성할 수 있으며, 악성코드 유포 네트워크를 분석하기 위한 기반이 될 수 있을 것으로 예상된다. 또한 멀티레벨 에뮬레이션은 페이지

하나를 분석하는 데 걸리는 시간이 1.29초로 상당히 빠른 속도로 분석할 수 있어 대량 홈페이지를 실시간으로 모니터링 하는 용도로 사용할 수 있을 것으로 기대된다.

향후 본 논문에서 제안한 멀티레벨 에뮬레이션의 결과 추출되는 링크 리스트의 특징을 추출하고 분석하는 기법에 대해서 보다 깊이 있는 연구가 필요할 것이다.

References

[1] Provos, N., McNamee, D., Mavrommatis, P., Wang, K., and Modadugu, N. "The ghost in the browser analysis of web-based malware," Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, pp. 4-4, Apr. 2007.

[2] Louis Marinos and Andreas Sfakianakis, "Threat landscape report", ENISA, Jan. 2013.

[3] Fahmida Y. Rashid, "Department of labor website hacked to distribute malware," "http://www.securityweek.com/department-labor-website-hacked-distribute-malware," May. 2013.

[4] Julianne Pepitone, "NBC hack infects visi tors in 'drive by' cyberattack," "http://money.cnn.com/2013/02/22/technology/security/nbc-com-hacked-malware/index.html," Feb. 2013.

[5] HAURI CERT 2Team, "Malware analysis report for Nateon hacking", HAURI, Aug. 2011.

[6] Graham Cluley, "DarkSeoul: Sophos-Lab s identifies malware used in south korean internet attack," "http://nakedsecu rity.sophos.com/2013/03/20/south-korea-cyber-attack/" March, 2013

[7] ASEC, "Malware analysis using 6.25 DDoS attack," Ahnlab, June. 2013.

[8] Ma, J., Saul, L.K., Savage, S., and Voelker, G.M., "Identifying suspicious URLs: an application of large-scale online learning," Proceedings of the 26th Annual International Conference on Machine Learning, pp. 681-688, Jun. 2009.

[9] Ma, J., Saul, L.K., Savage, S., & Voelker, G.M., "Beyond Blacklists: Learning to detect malicious web sites from suspicious URLs," Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1245-1254, Jun. 2009.

[10] Mavrommatis, N.P.P., and Monrose, M.A.R.F.., "All your iFRAMEs point to us," Usenix Security, pp. 1-15, Jul. 2008.

[11] Cova, M., Kruegel, C., and Vigna, G, "Detection and analysis of drive-by-download attacks and malicious Javascript code," Proceedings of the 19th international conference on World wide web, pp. 281-290, Apr. 2010.

[12] Chen, K.Z., Gu, G., Zhuge, J., Nazario, J., and Han, X.., "WebPatrol: Automated collection and replay of web-based malware scenarios," Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, pp. 186-195, Mar. 2011.

[13] Moshchuk, A., Bragin, T., Deville, D., Gribble, S.D., and Levy, H.M., "SpyProxy: execution based detection of malicious web content," Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, pp. 27-42, Aug. 2007.

[14] Wang, G., Stokes, J., Herley, C., and Felstead, D., "Detecting malicious landing pages in malware distribution networks," Proceedings of IEEE DSN, Jun. 2013.

[15] SpiderMonkey, "https://developer.mozilla.org/en-US/docs/Mozilla/Projects/SpiderMonkey"

〈저자소개〉

최 상 용(Sang-Yong Choi) 정회원
2000년 2월: 한남대학교 수학과.
2003년 2월: 한남대학교 컴퓨터공학과 공학석사
2012년 3월~현재: 한국과학기술원 사이버보안연구센터 선임연구원
2009년 3월~현재: 전남대학교 대학원 정보보안협동과정
〈관심분야〉 네트워크 보안, 악성코드, 해킹

강 익 선(Ik-Seon Kang) 정회원
2012년 2월: 제주대학교 컴퓨터공학과
2012년 9월~현재: 한국과학기술원 사이버보안연구센터 연구원
〈관심분야〉 시스템 및 네트워크 보안

김 대 혁(Dae-Hyeok Kim) 정회원
2010년 2월: 포항공과대학교 컴퓨터공학과
2012년 2월: 포항공과대학교 정보전자융합공학부 공학석사
2013년 6월~현재 : 한국과학기술원 사이버보안연구센터 연구원
2012년 3월~현재: University of Texas at Austin 컴퓨터 사이언스 박사 과정
〈관심분야〉 프로그래밍 언어, 웹 프로그래밍, 모바일 프로그래밍

노 봉 남(Bong-Nam Noh) 종신회원
1982년 2월: 한국과학기술원 전산학과 석사
1994년 2월: 전북대학교 대학원 전산과 박사
1983년~현재: 전남대학교 전자컴퓨터공학부 교수
2000년~현재: 전남대학교 시스템보안연구센터 소장
〈관심분야〉 정보보안, 디지털 포렌직, 시스템 및 네트워크 보안 등

김 용 민(Yong-Min Kim) 종신회원
2002년 2월: 전남대학교 전산통계학과 박사
2004년: 여수대학교 정보기술학부 전임강사
2006년~현재: 전남대학교 문화콘텐츠학부 부교수
〈관심분야〉 시스템 및 네트워크 보안, 전자상거래 보안 등