후킹 기법을 이용한 난독화 자바 스크립트 자동 해독 및 악성 웹 사이트 탐지 기술

오주형*, 임채태*, 정현철*
*한국인터넷진흥원
e-mail:{jhoh, chtim, hcjung}@kisa.or.kr

Automatic Javascript de-obfuscation and Detection of Malicious WebSite using Hooking Method

JooHyung Oh*, Chaetae Im*, HyunCheol Jung*
*Koreae Internet & Security Agency

요 약

무작위 SQL 삽입 공격 등을 통해 웹서버 해킹 사례가 꾸준히 증가하고 있으며, 대부분의 해킹된 웹서버는 난독화된 자바 스크립트 코드가 웹페이지에 삽입되어 악성코드 경유/유포지로 악용되고 있다. 본 논문에서는 난독화된 자바 스크립트 복원 및 취약한 ActiveX 생성에 사용되는 주요 함수에 대해 후킹 기술을 적용한 브라우저를 이용해서 난독화된 스크립트를 자동으로 해독하고, 악성코드 경유/유포지로 악용되는 웹 서버를 탐지할 수 있는 기술을 제안한다. 또한 제안 기술을 프로토타입 시스템으로 구현하고, 악성 URL 공유 사이트를 통해 수집한 난독화된 자바 스크립트 샘플 분석을 통해 제안한 기술이 높은 악성코드 경유/유포지 탐지율을 보이는 것을 증명한다.

1. 서론

기업이나 비영리 기관의 웹서버를 악성코드 감염을 위한 매개지로 악용하는 사례가 꾸준히 증가하고 있다. 악성코드 경유/유포지로 악용되는 웹서버의 대부분은 무작위 SQL 삽입 공격 등을 통해 악성 스크립트 코드가 특정 웹페이지에 삽입되어 있는 형태이며, 접속하는 사용자의 악성코드 감염을 유도하게 된다. 삽입된 악성 스크립트 코드는 악성코드 유포 사이트로의 리다이렉션 코드와 악성코드 감염을 위한 익스플로잇 코드로 나눌 수 있다. 주로 사용되는 리다이렉션 코드는 높이 0, 넓이 0의 hidden iframe을 통해서 악성코드가 있으며, 익스플로잇 코등의 경우 취약한 ActiveX 생성 스크립트 코드와 공격코드로 이루어져 있다.

취약한 웹 서버의 특정 페이지에 삽입되어 있는 악성 스크립트는 시그니처 기반 악성 웹 서버 탐지 시스템을 통해서 탐지할 수 있다. 그러나 최근 시그니처 기반 탐지 회피를 위해 인코딩, 암호화, 동적 스크립트 생성 등을 이용한 다양한 난독화 기술이 사용됨에 따라 탐지에 어려움이 있다. MS사의 2008년 보고 자료에 따르면, 삽입된 악성 자바 스크립트 코드의 90%이상이 난독화 기술이 적용되어 있으며, 최근 다양한 난독화 패턴이 적용되어 분석 및 탐지가 더욱 어려운 실정이다. 따라서 본 논문에서는 난독화된 스크립트 자동 해독이 가능하고, hidden iframe을 통한 악성코드 경유와 취약한 ActiveX 공격을 통한 악성코드 감염을 유도하는 악성코드 유포지를 탐지 할 수 있는 기술을 제안한다. 난독화된 악성 자바 스크립트는 사용자의 인터넷 브라우저에서 최종적으로 악성코드 유포지로의 리다이렉션 또는 공격 코드를 복원하게 된다. 본 논문에서는 난독화된 악성코드가 복원되는 시점에 후킹 기술을 적용해서 오리지날 악성코드를 탐지 할 수 있는 기술을 제안한다. 또한, 악성코드 경유지로부터 추출한 웹 사이트 정보를 이용해서 취약한 ActiveX 객체를 생성하는 시점에 후킹을 걸어서 익스플로잇 사이트를 탐지할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서 난독화된 자바 스크립트 해독 기술과 악성 웹 사이트 탐지 기술을 설명하고, 3장에서 제안하는 난독화 해독 및 악성 웹 사이트 탐지 기술을 제안한다. 4장에서 악성 자바스크립트 샘플을 이용한 악성 웹 사이트 탐지 성능분석 결과를 설명하고, 5장에서 결론 및 향후 연구에 대해 기술한다.

2. 관련 연구

웹 해킹 공격을 통해 정상적인 웹서버를 악성코드 경유/유포지로 악용하는 사례가 꾸준히 보고됨에 따라, 웹사이트를 주기적으로 점검해서 악용되는지 점검하고, 유포되는 악성코드를 수집할 수 있는 다양한 기술이 연구되고 있다. YoungHan CHoi는 [1]에서 자바 스크립트의 스트링 패턴 분석을 통해 난독화된 자바스크립트를 포함하고 있는 악성 웹 사이트 탐지 기술을 제안하였다. 난독화된 자바 스크립트의 경우, 긴 문자열 포함, 특수 문자의 반복적인 사용 등의 특징을 포함하고 있으므로, 이와 같은 패턴

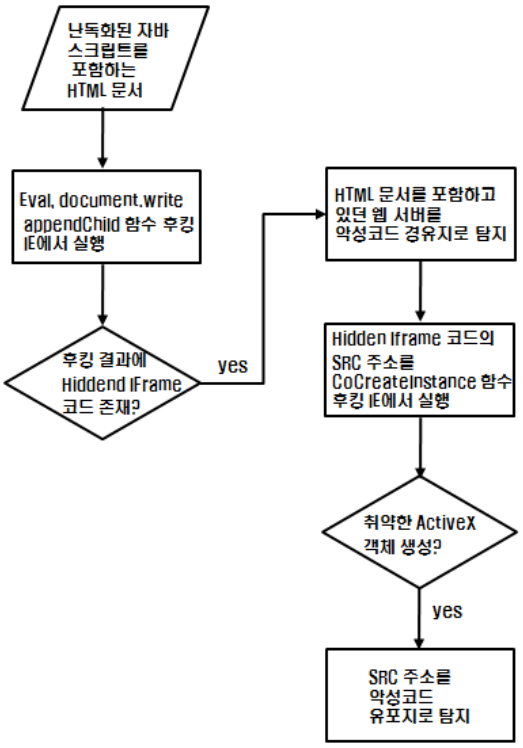
분석을 통해 난독화된 자바 스크립트 여부를 판단하고, 이를 분석할 수 있는 기술을 제안하였다. YoungHan CHoi가 제안한 기술은 의심스러운 자바 스크립트 실행 없이 스트링과 문자 패턴 분석을 통해 난독화 여부를 판단하고 있으므로, 행위 기반 분석 기술에 비해 빠른 속도로 탐지가 가능하다. 그러나, 금융 결제, 전자 상거래 등에 사용되는 중요한 스크립트 보호등에 난독화 기술이 사용되고 있으므로, 난독화된 스크립트를 포함하고 있는 웹서버를 무조건 악성으로 판단하기에는 오탐이 존재한다. 또한, 새로운 난독화 패턴이 나올 경우 난독화 여부를 판단하지 못하는 단점이 있다.

Andreas Dewald는 [x]에서 자바 스크립트 샌드 박스를 이용한 난독화 자바 스크립트 해독 및 악성 웹 사이트 탐지 기술 ADSandbox을 제안하였다. ADSandbox는 내부적으로 오픈소스 기반의 자바 스크립트 실행 엔진 SpiderMonkey를 포함하고 있어, 인터넷 브라우저 없이 자바 스크립트 실행 및 출력 결과 확인이 가능하다. 따라서, 의심 URL로부터 응답 HTML 문서를 다운로드 받은 다음 의심스러운 자바 스크립트 블락들을 추출하여, 실행하는 구조를 채택하고 있다. 그러나, 자바 스크립트 블락들이 인터넷 브라우저에서 제공하는 HTML 코드 또는 DOM 코드들을 포함하고 있기 때문에 모든 자바 스크립트 블락에 대해서 실행이 불가능한 단점이 존재한다.

Feinstein은 난독화된 자바 스크립트 해독을 위한 Caffeine Monkey를 제안하였다. 그는 ADSandbox의 접근 방식과 비슷하게 SpiderMonkey를 이용하여, 난독화된 자바 스크립트를 실행하는 구조를 가지고 있다. 차이점은 SpiderMonkey를 수정해서 자바스크립트 eval 함수와 스트립 결합 함수에 후킹 기법을 적용하여, 난독화된 자바 스크립트가 복호화되는 시점에 악의적인 원본 스크립트 코드를 얻을 수 있게 된다. 그러나, ADSandbox와 같이 인터넷 브라우저에서 제공하는 HTML 코드 또는 DOM 코드들을 포함한 자바 스크립트 실행에 어려움이 있다.

3. 제안하는 악성코드 경유/유포지 탐지 기술

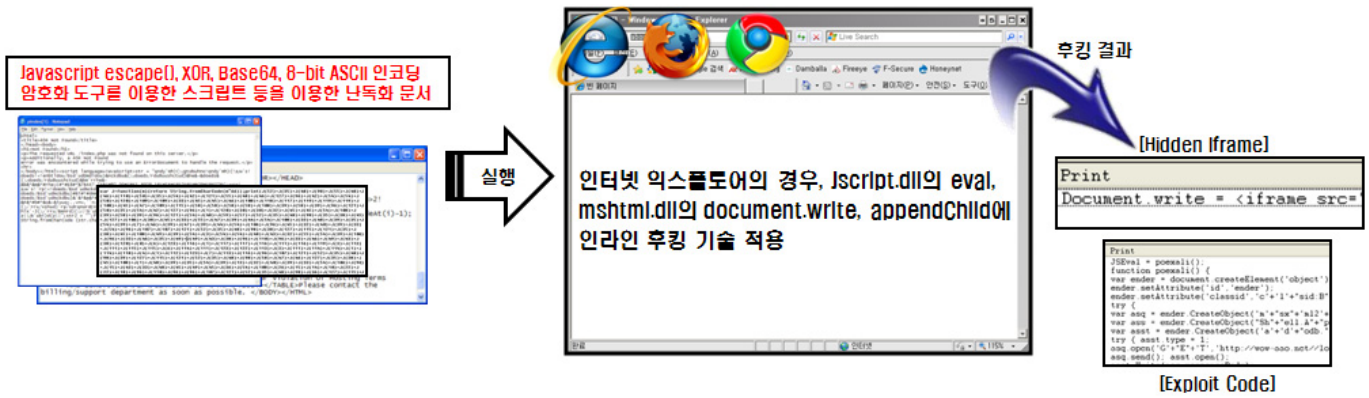
제안하는 악성코드 경유/유포지 탐지 기술은 [그림 1] 같은 방식으로 동작한다. 난독화된 자바 스크립트 코드를 포함하는 HTML 문서를 입력으로 입력 받아, jscript.dll의 eval 함수, mshtml.dll의 doccument.write 함수 및 element.appendChild 함수에 in-line 후킹 기술이 적용된 브라우저 실행한 다음 후킹 결과로 출력되는 Hidden IFrame이 있을 경우 악성코드 경유지로 탐지한다. 또한 Hidden Iframe의 SRC 주소를 CoCreateInstance 함수에 후킹 기술이 적용된 IE에서 실행한 다음 후킹 결과에 포함된 CLSID 값이 취약한 ActiveX ID 일 경우 악성코드 유포지로 탐지 하게 된다.



[그림 1] 제안하는 악성코드 경유/유포지 탐지 기술 흐름도

[표 1] 난독화 스크립트 해독 및 악성 웹 사이트 탐지 기법 비교

	난독화 스크립트 해독	악성 웹 사이트 탐지	속도	기타
YoungHan CHoi 기법	해독 불가 (탐지만 가능)	기능 제공	빠름 (높은 오탐 및 미탐 발생)	
ADSandbox	해독 가능	기능 제공	느림 (상대적으로 낮은 오탐)	빈번한 자바스크립트 실행 오류 발생
Caffeine Monkey	해독 가능	N/A(Self)	느림 (상대적으로 낮은 오탐)	빈번한 자바스크립트 실행 오류 발생



[그림 2] 난독화 스크립트 자동 해독 기술

3.1 동적 출력 함수 후킹을 통한 난독화된 스크립트 자동 해독 및 악성코드 경유지 탐지

자바 스크립트 난독화 기법은 코드 인코딩, 암호화 등의 기법을 이용해 악성코드 유포 사이트로의 리다이렉션 코드, 악성코드 감염을 위한 악성 익스플로잇코드를 숨기는 것을 목적으로 하고 있다. 또한 난독화된 스크립트 코드는 최종적으로 사용자의 인터넷 브라우저에서 eval, document.write, appendChild 등의 함수를 이용해서 동적으로 악의적인 스크립트를 복원하게 된다. 따라서, 난독화된 스크립트 코드는 사용자의 인터넷 브라우저에서 복원되는 시점에 후킹을 걸어서 자동 해독이 가능하다. 본 논문에서는 인터넷 익스플로워의 자바 스크립트 실행을 담당하는 jscript.dll의 Eval() 함수, 문서(Document) 객체 처리를 담당하는 mshtml.dll의 document.write() 함수와 appendChild() 함수에 인라인 후킹 기술을 적용해서 악의적인 스크립트가 복원되기 직전에 획득할 수 있게 된다.

Detour 라이브러리를 이용한 인라인 함수 후킹 기술은 원본 함수의 코드 리라이팅 기술을 통해 트램블린 함수로 호출을 리다이렉션 한 다음 입력 파라미터를 후킹하는 기법이다.

jscript.dll(v5.7.0)의 eval 함수의 경우, 인터넷 익스플로어가 실행 된 이후 메모리상의 437350EA에 로딩되기 때문에 해당 메모리 번지에 우리가 생성한 DetouredJsEval 함수로 점프 하는 코드를 삽입하게 된다. 따라서, 브라우저에서 실행되는 JSEval 호출은 DetouredJsEval 함수로 리다이렉션 되게 되고, String 형태의 입력 해독 스트링을 얻을 수 있다. 따라서, 그림 1에서 보는 것과 같이 Eval, document write 등을 통해서 동적으로 생성되는 생성되는 Hidden Iframe과 같은 해독 코드를 얻을 수 있다. 대부분의 악성 웹 사이트는 악성코드 감염 사이트의 존재를 숨기기 위해 악성코드 경유지를 두고 있으며 악성코드 경유지에서 악성코드 공유 사이트로의 연결을 위해 hiiden iframe을 포함하고 있다. 따라서, 난독화 기술을 이용해서 hidden Iframe을 포함하고 있는 웹서버는 악성코드 경유지로 탐지 되게 된다. hiiden iframe은 다음 4가지 패턴을 기준으로 탐지하게 된다.

- <iframe src=A.COM width=0 height=0></iframe>
- <iframe src=A.COM width=1 height=0></iframe>
- <iframe src=A.COM width=0% height=1%></iframe>
- <iframe src="A.COM" width=150 height=100 style="display:none;"></iframe>

3.2 COM 객체 생성 함수 후킹을 통한 악성코드 유포 서버 탐지

스크립트 난독화 해독 기술을 통해 탐지된 Hidden Iframe의 목적지 주소에 대해서 CoCreateInstance 함수 후킹 기술이 적용된 브라우저에서 다시 한번 실행하게 된다. CoCreateInstace 함수는 Com 객체 생성을 위해 최종적으로 호출되는 함수로 인터넷 익스르롤러에서 액티브A 생성 시 호출 된 경우 입력파라미터로 CLSID 정보를 포함하게 된다. 따라서, CoCreateInstance 함수에 후킹 기술을 적용할 경우 생성하는 ActiveX를 탐지할 수 있다. 생서되는 AcitveX ID 정보는 마이크로소프트사에서 제공하는 killbit 리스트와 비교하게 된다. killbit 리스트는 취약점이 보고된 특정 버전의 ActiveX 정보를 포함하고 있다. 따라서 악성코드 경유지에서 리다이렉션 되는 사이트가 취약한 ActiveX 객체를 생성을 시도할 경우 악성코드 유포 사이트로 탐지하게 된다.

4. 성능 평가

본 논문에서 제안하는 후킹 기법을 이용한 난독화 자바 스크립트 자동 해독 및 악성 웹 사이트 탐지 기법의 성능 검증을 위해 2010년 7월부터 9월까지 malwaredomainlist, .blade-defender 등과 같은 악성 URL 공유 사이트에서 수집된 193개의 웹페이지 URL 샘플을 이용하여, 난독화된 자바 스크립트 해독 및 악성 웹사이트

탐지 분석 기능을 평가하였다. 또한, Heritrix Crawler를 이용하여, 수집된 악성 웹페이지로부터 응답 HTML 문서를 다운로드 받아, 동적 스크립트 출력 함수 및 COM 객체 생성 함수에 대한 후킹 기술이 적용된 인터넷 익스플로러에서 실행한 다음, 결과를 다음과 같이 분석하였다.

1. http://sharonsten---/p7/index.php 에서 HTML 문서 다운로드
2. 후킹 기술이 적용된 IE에서 HTML 문서 실행
3. 후킹 결과 분석
   - CLASSID: CA8A9780-280D-11CF-A24D-444553540000를 가지는 취약한 ActiveX 생성 탐지
4. 난독화가 해독된 악성 스크립트 분석
   - Document.write 함수 후킹 결과 분석

```
<body><div id='j'></div><OBJECT id=Pdf1 height=0 width=0
classid=clsid:CA8A9780-280D-11CF-A24D-444553540000>
</OBJECT><style type='text/css'>.css {behavior:
url(#default#userData);}</style><MARQUEE id='mrq' class=
'css'></MARQUEE><iframe src='' frameborder='0' width='1'
height='1' id='hello' name='hello'></iframe></body>
```

   - 취약한 ActiveX 생성 확인 및 페이지 리다이렉션 탐지

분석 결과 성능 [표 2]에서 보는 것과 같이, 수집된 193개의 웹사이트 URL 샘플 리스트로부터, 120개의 URL이 악성으로 탐지되었다.

[표 2] 성능 평가 결과

전체 URL	193개
중 악성 URL	120개
중 난독화된 자바 스크립트 포함	120개
중 취약한 ActiveX 생성	80개
중 리다이렉트 코드 포함 URL	60개
중 정상 URL	73개
중 악성스크립트 코드 없는 URL	40개
중 HTML 문서 실행 오류	33개

120개 악성 URL 중 Adobe Acrobat PDF ActiveX의 취약점 공격 익스플로잇 코드등다양한 악성 스크립트 코드를 동적으로 생성하는 80개의 URL을 탐지하였으며, 120개의 악성 URL 중 60개의 웹사이트에서 외부 도메인으로의 페이지 리다이렉션을 유도하는 hidden IFrame이 존재하는 것을 탐지하였다.

그리고, 193개의 샘플 URL 중 정상 탐지된 73개 URL을 선별하여, HTML 문서 상세 분석을 수행한 결과 문서 내에 난독화된 스크립트 코드가 존재하지 않는 40개의 정상 웹사이트가 포함되어 있음을 분석하였다. 또한, 33개의 웹사이트는 해커에 의한 웹 해킹과정에서 HTML코드 또는 정상 자바 스크립트 코드의 일부가 삭제되어 정상적인 실행이 되지 않는 것을 확인하였다.

5. 결론 및 향후 계획

최근 악성 봇넷을 이용한 무작위 SQL 삽입 공격이 급증하고 있으며, 삽입된 SQL을 통한 웹서버 해킹 사례가 지속적으로 보고되고 있다. 본 논문에서는 난독화된 자바 스크립트가 악성 스크립트 복원을 위해 사용하는 주요 함수에 후킹 기술을 적용한 브라우저에서 샐행 한 다음 난독화된 스크립트를 해독하고, 악성 웹사이트 URL을 탐지할 수 있는 있는 기법을 제안하였다. 제안하는 알고리즘은 자바 스크립트를 웹 브라우저에서 실행하기 때문에 기존 분석 기법에 비해 높은 탐지율과 낮은 오탐율을 보장 받을 수 있으나, 자바 스크립트 패턴 분석 등과 같은 정적 분석 기법에 비해 더 많은 분석 시간이 필요한 단점이 존재한다. 따라서, 향후 악성 웹사이트 URL의 전처리 분석에 정적 분석을 활용하는 하이브리드 형태의 분석 기법에 대한 연구가 필요하다.

Acknowledgment

본 연구는 지식경제부 및 한국산업기술평가관리원의 IT 산업원천기술개발사업의 일환으로 수행하였음. [10035427, 지능형 악성코드 자동 분석 및 경유/유포지 탐지 기술 개발]

참고문헌

[1] YoungHan Choi, TaeGhyoon Kim, SeokJin Choi and CheolWon Lee, " Automatic Detection for JavaScript Obfuscation Attaccks in Web Pages through String Pattern Analysis", FGIT 2009,

[2] Andreas Dewald, Thorsten Holz, Felix C. Freiling, " ADSandbox: Sandboxing JavaScript to Fight Malicious Websites", In proceeding of the 25th Symposium On Applied Computing (SAC 10), March 2010

[3] Ben Feinstein, Daniel Peck, "Caffeine Monkey: Automated Collection, Detection and Analysis of Malicious JAvaScript", Black Hat USA, 2007

[4] Galen Hunt, Doug Brubacher, "Detours: Binary Interception of Win32 Functions", USENIX 99