난독화 된 자바스크립트 탐지를 위한 실행 단위 코드 기반 특징 추출에 관한 연구

A Study on Feature Extraction for Detection for Obfuscated Javascript Based on Executable Code Units

저자
(Authors)

강익선, 조호묵
Ik-Seon Kang, Ho-Mook Cho

출처
(Source)

한국컴퓨터정보학회 학술발표논문집 22(2), 2014.7, 79-80 (2 pages)
Proceedings of the Korean Society of Computer Information Conference 22(2), 2014.7, 79-80 (2 pages)

발행처
(Publisher)

한국컴퓨터정보학회
The Korean Society Of Computer And Information

URL

http://www.dbpia.co.kr/Article/NODE06603110

APA Style

강익선, 조호묵 (2014). 난독화 된 자바스크립트 탐지를 위한 실행 단위 코드 기반 특징 추출에 관한 연구. 한국컴퓨터정보학회 학술발표논문집 , 22(2), 79-80.

한국컴퓨터정보학회 하계학술대회 논문집 제22권 제2호 (2014. 7)

난독화 된 자바스크립트 탐지를 위한 실행 단위 코드 기반 특징 추출에 관한 연구

강익선°, 조호묵*
°*한국과학기술원, 사이버보안연구센터
e-mail:{ikseon1026,chmook79}@kaist.ac.kr*°

A Study on Feature Extraction for Detection for Obfuscated Javascript Based on Executable Code Units

Ik-Seon Kang°, Ho-Mook Cho*
°*Cyber Security Research Center, KAIST

● 요 약 ●

악성코드 유포를 위해 가장 많이 사용되는 Drive-by-download 공격에는 주로 자바스크립트가 사용되며, 공격자는 탐지 시스템의 우회하고 행위 분석을 어렵게 하기 위해 공격에 사용되는 스크립트를 난독화 한다. 난독화 된 자바스크립트를 탐지하기 위한 여러 기존의 연구들이 있었지만 최소한의 코드가 난독화 되어 있거나 정상 코드와 혼재할 경우 난독화 여부를 판단하기 어려운 한계가 있다. 본 논문에서는 난독화 된 자바 스크립트를 효과적으로 탐지하기 위해 전체 스크립트를 실행 단위 코드로 나눠 분석에 필요한 특징을 효과적으로 추출하는 방법을 제안한다.

키워드: 자바스크립트(Javascript), 난독화(Obfuscation)

Ⅰ. 서 론

악성코드를 유포하는 방법으로 잘 알려진 Drive-by-download 공격에는 자바스크립트가 주로 사용된다. 공격자는 시그니처 기반 탐지 시스템의 탐지를 우회하고 행위 분석을 어렵게 하기 위해 공격에 사용되는 악성 스크립트를 난독화 한다. 이러한 공격에 대응하기 위하여 난독화 된 자바스크립트와 악성코드 유포지 탐지에 대한 관련 연구가 진행되어 왔다[1,2]. 하지만 최소한의 코드를 난독화 하거나 난독화 된 코드가 정상적인 코드와 함께 존재 할 경우에 정상적인 코드의 특징으로 인해 코드의 난독화 여부를 판단하기 어렵다. 따라서 기존 연구의 한계점을 극복하며, 난독화 된 자바스크립트를 효과적으로 탐지하기 위해 실행 단위 코드의 특징을 추출하는 방법을 제안한다.

Ⅱ. 배경 지식

자바스크립트는 객체 기반의 스크립트 언어로 웹브라우저에서 실행된다. HTML의 DOM 객체뿐만 아니라 다른 응용프로그램에 내장된 객체에도 접근 할 수 있어 공격자는 악성코드를 유포 할 때 자바스크립트를 많이 사용한다. 공격자는 탐지 시스템을 우회하며, 악성 행위의 분석을 숨기기 위해서 다양한 자바스크립트 난독화 기술을 사용한다. 많이 사용되는 난독화 방법에는 문자열 분리, 치환, 16진수 또는 유니코드 등의 인코딩과 같은 문자열 형태 변환, eval(), unescape(), document.write() 등의 함수의 사용이 포함된다. 뿐만 아니라 자바스크립트 언어의 변수가 숫자, 스트링 및 배열 등 여러 가지 형태로 변환될 수 있는 점과 생성자 및 에러 메시지를 사용하여 변환 될 수 있는 점이 난독화 기법에 사용된다.

Ⅲ. 실행 단위 코드의 특징 추출

본 논문에서 제안 하는 실행 단위 코드의 특징을 추출하기 위해서는 전체 스크립트를 단위 블록으로 나눠야 한다. 실행 단위 블록이란 자바스크립트 인터프리터에서 실행 가능한 단위 코드를 의미하는 것으로 함수나 변수의 정의 조건문 또는 반복문 등이 여기에 속한다. 실행 단위 코드의 난독화 여부를 판단하기 위해 아래와 같은 특징을 정의하였다.

- Feature 1. \x 문자의 비율

문자열이 16진수로 난독화 되어 있는 코드를 탐지하기 위한 특징이다.

- Feature 2. 특수문자의 비율

변수 타입의 문자열 및 숫자, 스트링 및 배열 등 여러 형태로 자유롭게 변환이 가능하다는 자바스크립트의 기능을 이용하여 난

독화 한 경우를 탐지하기 위한 특징이다.

- Feature 3. eval 파라미터의 비율

난독화 된 자바스크립트를 복호화하기 위해서 사용되는 함수로, 해당 함수로 전달되는 파라미터의 비율을 탐지하며 난독화 여부를 판단하기 위한 특징이다.

- Feature 4. unescape 파라미터의 비율

escape() 함수로 인코딩된 문자열을 디코딩하는데 사용되는 함수로 난독화 된 코드를 복호화 하는데 사용 된다는 점을 이용하여 난독화 여부를 판단하기 위한 특징이다.

- Feature 5. document.write 파라미터의 비율

document.write() 함수는 난독화 된 코드가 복호화 되어 현재 파싱중인 문서에 내용을 쓰는데 사용 될 수 있다. 이 점을 이용하여 난독화 여부를 판단하는데 사용되는 특징이다.

Ⅳ. 실험

실험의 목적은 난독화 스크립트 탐지를 위해 스크립트 코드 블록 단위을 나누고, 해당 블럭에서 특징을 추출할 수 있는지 검증하는 것이다. 이를 위하여 Firefox 브라우저에서 사용되고 있는 자바스크립트 엔진인 Spidermonkey[3] 가 사용되었다. 전체 스크립트를 실행 단위 코드로 나누기 위해 Spidermonkey의 API 중 JSBool JS_BufferIsCompilableUnit(JSContext *cx, JSObject *obj, const char *bytes, size_t length)을 사용하였다.

표 1. 스크립트로부터 추출된 특징 값
Table 1. Feature values

난독화 여부	F1	F2	F3	F4	F5
○	0.0174	0.065	0.000	0.000	0.113
○	0.000	0.200	0.565	0.000	0.004
○	0.000	0.217	0.300	0.000	0.104
○	0.000	0.220	0.297	0.000	0.098
○	0.000	0.271	0.000	0.381	0.170
×	0.000	0.076	0.000	0.000	0.000
×	0.000	0.098	0.000	0.000	0.000
×	0.000	0.176	0.000	0.000	0.000
×	0.000	0.195	0.000	0.000	0.000
×	0.000	0.093	0.000	0.000	0.000

난독화 된 스크립트 5개와 정상 스크립트 5개를 대상으로 실행 단위 코드 특징 추출을 수행 했으며 실험 결과는 표1에 나타나 있다. 실험 결과 성공적으로 실행 단위 코드 블록을 나눌 수 있었고, 정의 된 특징들을 각각의 단위 스크립트에서 추출 할 수 있었다.

Ⅴ. 결론

본 논문에서는 난독화 된 스크립트를 효과적으로 탐지하기 위해 실행 단위 코드 기반으로 스크립트에 존재하는 특징을 추출할 수 있는 방법을 제안하였다.

실험을 통하여 전체 코드를 실행 단위 코드로 나눌 수 있음을 보였고, 난독화 여부를 판단하는 지표로 활용 되는 특징들을 성공적으로 추출 할 수 있음을 확인 하였다.

본 연구를 바탕으로 난독화 된 스크립트를 탐지 할 수 있을 뿐만 아니라 악성코드 유포 페이지 탐지에 대한 연구에 활용 될 것으로 기대한다.

참고문헌

[1] Marco Cova, Christopher Kruegel, and Giovanni Vigna, "Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code," International World Wide Web Conference Committee(IW3C2), April 2010.

[2] YoungHan Choi, TaeGhyoon Kim, SeokJin Choi, and CheolWon Lee, "Automatic Detection for JavaScript Obfuscation Attacks in Web Pages through String Pattern Analysis," FGIT 2009, pp. 160.172, December 2009

[3] Mozilla, SpiderMonkey, https://developer.mozilla.org/ko/docs/SpiderMonkey