난수 및 해시함수를 이용한 자바스크립트 난독화 방안 연구

[1] 김영호, [1] 최동식, [2*]여상수
[1] 목원대학교 융합미디어컴퓨터학부, 학사과정,{ kimho5291, opert1564}@naver.com
[2*] 목원대학교 융합미디어컴퓨터학부, 교수, sangsooyeo@gmail.com

A Research of JavaScript Obfuscation Using Random Numbers and Hash Function

[1]Young-Ho Kim, [2]Dong-Sik Choi, [31]Sang-Soo Yeo
[1] Division of Convergence Computer& Media, Mokwon University, Bachelor Candidate, {kimho5291, opert1564}@naver.comr
[2*] Division of Convergence Computer& Media, Mokwon University, Professor, sangsooyeo@gmail.com

요 약

월드와이드웹은 지금 이 시대의 모든 인터넷 자원을 표현하는 기반으로 활용되고 있다. 이러한 시대적 요구에 따라 웹 서버와 웹 기술은 지속적으로 발전하고 있다. 웹 서비스는 서버 측과 클라이언트 측으로 나누어져서 각각의 영역에서 많은 기술 개발이 이루어지고 있다. 서버 측에서 실행되는 코드들은 서버 내에서 실행된 후 그 결과만 전송되기 때문에, 코드보호가 상대적으로 쉬운 반면, 클라이언트 측에서 실행되는 코드들은 클라이언트 웹 브라우저에서 실행되어야 하기 때문에 코드 자체가 클라이언트로 넘어가야 하는 특수한 상황이다. 그럼에도 불구하고, 클라이언트 측에 대한 보안 방안에 대한 연구개발은 많이 이루어지지 않았다. 본 논문에서는 웹 클라이언트 측 보안 강화를 위해 자바스크립트 코드를 난독화하는 방안에 대한 연구를 진행하였다. 기존의 난독화 기법은 정적이며 역공학을 통해 분석해 내거나 역함수를 만들어내는 것이 상대적으로 쉬웠다. 본 논문에서 제안하는 난독화 기법은 소스 코드의 복잡도를 높이기 위해 프로그램 로직을 일부 변경하는 것과 난수와 해시 함수를 사용하여 난독화를 주기적으로 실행하게 된다. 이를 통해서 로직 분석을 어렵게 하고, 역공학 과정을 상대적으로 어렵게 만든다. 결국 제안한 난독화 기법은 공격에 들어가는 비용과 시간을 증가시킬 수 있으며, 역함수 프로그램을 만들기 어렵게 할 수 있다.

Abstract

The World Wide Web is now used as the foundation for all the Internet resources of this era. In response to the demands of the times, Web servers and Web technologies have been continuously developing. Actually, Web services are divided into server side and client side, and many technologies are developed in each area. Code protection on the server side is relatively easy because the codes are executed on the server side, and only the result HTML codes are transmitted to the client side. On the other hand, the codes on the client side have to be executed on the web browser at the client side. This is not a common situation in terms of security. Nevertheless, much research and development on security measures on the client side has not been carried out until now. This paper investigated how to obfuscate JavaScript codes to enhance the client-side security. The existing obfuscation techniques are almost static and relatively easy to analyze through reverse engineering, and also easy to produce

* Corresponding Author
Received: Dec. 12, 2019, Revised: Dec. 31, 2019, Accepted: Dec. 31, 2019

inverse function codes. This paper proposes a novel obfuscation scheme, which periodically changes the program logic to increase the complexity of the source codes and periodically executes the obfuscation process using random numbers and hash functions. This makes logic analysis more difficult and makes the reverse engineering process more difficult than the existing schemes. As a result, the proposed obfuscation scheme can increase the cost and time of attack, and make it difficult to make an inverse function program.

Keywords: *Obfuscation, Logic Configuration, Hash Algorithm, String Processing, String Analysis, Security, Code Security, Client-side Web Security*

I. 서론

기술의 발달에 따라 해킹 공격의 위협도 늘어나고 있다. 특히 '안랩, 2019 년 3 분기 사이버 공격 동향 발표' 에 따르면 웹 기반 공격이 44%로 가장 높은 비중을 차지하고 있다. 또한 '2014 웹 공격 동향 보고서, 펜타시큐리티 시스템' 에 의하면 취약성 파악이 38.1%로 가장 높았고, 정보유출이 20.4%로 뒤를 따르고 있다. 웹 서비스의 중요성은 커지고 있는 반면, 웹 서비스에는 다양한 취약점이 존재하며 정보 유출과 취약점 파악에 관한 보안 관리는 허술한 편이다.

웹 보안은 서버 측 보안과 클라이언트 측 보안으로 나눠진다. 클라이언트 측은 웹 서비스 특성상, 코드가 노출된다. 또한 웹 인터셉트 프로그램을 사용 시, 입력 값 변조 및 코드 검증 무효화가 가능하다[1]. 그렇기 때문에 서버 측에서 한 번 더 입력 값 검증을 수행해야 한다. 이러한 문제점 때문에 클라이언트 측보다 서버 측 보안을 더 중요시하고, 클라이언트 측 보안은 서버 측 보안에 비하여 관심이 적은 편이다. 그러나 클라이언트 측에서 노출된 코드를 분석할 경우, 서버 측 코드와 로직을 예측 및 분석하거나 직접적인 공격 또한 가능하기 때문에 클라이언트 측 보안 또한 서버 보안 못지않게 중요하다고 볼 수 있다.

본 연구에서는 이러한 문제점을 보완할 수 있는 클라이언트 측 보안을 강화하는 방법으로 난독화(obfuscation) 기법을 제시한다. 난독화는 코드를 사람이 읽거나 분석하기가 어렵도록 만드는 과정을 의미한다. 난독화를 하더라도 웹 브라우저는 난독화된 자바스크립트 코드를 여전히 실행할 수 있어야 한다. 난독화와 관련된 기존 연구 또는 기존 난독화 코드들은 존재하지만, 본 연구에서는 기존 기술들의 한계를 식별하고, 그에 대한 대책을 제시한다. 본 연구의 초기 연구결과는 [2] 및 [3]으로 학술 발표를 진행하였고, 최종적인 연구결과는 본 논문으로 제시한다.

II. 관련 연구

현재까지 사용되고 있는 난독화 기법으로는 코드 압축, 더미코드 삽입, 인코딩(Hex, ASCII) 등이 있으며, 추가적인 난독화 처리 기법으로는 Split 기법, Hexdecimal() 함수, XOR 등 혼용하여 사용되고 있다[4].

- 코드 압축(code compression) : 자바 스크립트 및 HTML 코드에 존재하는 모든 공백문자들을 단순화하여 단어 별 띄어쓰기를 하나의 공백 문자로 치환하는 방법이다. 특히 자바스크립트의 세미콜론(;) 이나 중괄호({, })등은 줄 바꿈으로 보통 이어지지만, 코드압축 방법을 거치면 세미콜론 또는 중괄호 뒤에는 바로 다음 줄의 코드가 이어져서 기술된다.

- 더미코드 삽입(inserting dummy codes) : 자바 스크립트 코드 내에 의미가 없는 더미코드를 추가함으로써, 코드의 가독성을 떨어뜨리는 기법이다. 더미코드가 추가되더라도 코드의 실행에는 문제가 없어야 한다.

- 인코딩 변경(Altering codes) : 자바스크립트 코드를 다른 종류의 코드로 인코딩하는 기법이다. 영문 알파벳들을 BASE64 코드 또는 HEX 코드 등으로 인코딩하고 eval() 함수 등을 통해서 이를 실행하는 방법이다.

이러한 기술들은 상당 기간 동안 사용되어온 난독화 기법들이기 때문에 많이 알려져 있으며, 쉽게 역공학이 가능한 기법들이라고 볼 수 있다. 많은 시간을 들이지 않고 난독화 이전의 소스코드를 획득할 수 있기 때문에 클라이언트 측 코드의 보안을 높여주기에는 다소 부족하다고 볼 수 있다. 적절한 시간만 투입한다면 되기 때문에 난독화로서 의미가 사라진다.

III. 자바스크립트 난독화 연구

1. 자바스크립트 난독화 방안

웹 서비스와 클라이언트 측 보안을 위한 연구로서 자바스크립트에 관한 난독화를 수행한다. 본 연구에서는 키가 존재하며 복호화가 가능한 일반적인 형태의 기법이 아닌, 단방향성을 가진 해시 함수를 이용한 난독화와 코드를 난잡하게 만드는 로직 변경, 그리고 일정한 시간마다 난독화를 수행하는 방법으로 연구를 진행한다. 난독화 규칙은 프로그램이 수행할 때마다 바뀌며 복호화 관련 키는 존재하지 않는 방향으로 접근한다. 사용자는 원본 디렉터리에서 파일을 수정하게 된다면 웹 동작이 되는 디렉터리에 새롭게 난독화가 수행된 파일이 저장된다.

여기서 난독화는 로직 변경과 해시 함수를 사용하므로 어느 정도의 수행시간이 존재한다. 파일을 순차적으로 난독화 하기 때문에 수행되는 시간에 웹이 동작된다면 많은 문제가 일어날 것이다. 그러므로 두 개 이상의 디렉터리를 생성하여 하나의 디렉터리가 웹에서 동작할 때, 다른 디렉터리에서는 새로운 난독화 파일 생성을 수행한다. 일정한 시간을 정해서 두 개의 디렉터리 포인터를 번갈아 가면서 난독화 디렉터리를 가리키게 구현한다.

난독화 수행 시, 적용되지 않는 부분도 고려해야 한다. 난독화로 변하지 않아야 할 값은 자료형, 예약어, 키워드, 인용구(‘’, " ") 등의 프로그램이 인식해야 하는 토큰은 변경되지 않게 주의한다. 또한 자바스크립트나 html 안에 있는 스크립트 같은 경우, 다른 디렉터리에 있는 함수를 호출할 수가 있다. 이러한 경우에는 난독화를 수행할 디렉터리를 모두 탐색하여 함수명을 수집 후, 예외처리를 하거나 동일한 난독화 규칙을 만들어서 수행한다. 본 연구에서는 난독화가 적용될 디렉터리만 난독화가 수행되고 그 외, 외부에서 접근하는 자바스크립트의 함수 같은 경우는 예외로 처리한다[5].

3.2. 연구 범위

외부 디렉터리를 호출하는 경우, 그에 따른 함수, 클래스의 명을 알 수 없기 때문에 코드 매칭을 할 수 없다. 예를 들어 CDN(Content Delivery Network)을 사용하는 경우에는, 정해진 서버에 접속하는 것이 아니라 사용자와 가장 가까운 서버에 접속하도록 유도하기 때문에, JQuery 나 리덕스 등을 직접 파일을 저장하지 않고 사용이 가능하다. 본 연구에서는 함수명을 난독화 하기 때문에 외부 함수를 자바스크립트에서 사용된다면, 이것 역시 난독화를 수행하게 되고 실행 시, 치명적인 오류가 발생한다. 문제 해결 방안으로 외부 파일까지 검색하여 예외처리를 하는 방법이 있지만, 외부 파일에서도 또 다른 외부 파일을 호출할 수 있으므로, 많은 변수와 실행시간이 들기 때문에 실질적으로 불가능하다. 그렇기 때문에 본 연구에서의 범위는 내부 디렉터리 난독화로 한정한다.

3.3. 프로그램 설계

내부 디렉터리를 탐색하고 난독화 수행하는 방법으로, DFS(Depth-First Search) 탐색 알고리즘으로 가장 내부에 있는 디렉터리를 탐색하여 HTML 과 자바스크립트 파일을 찾는다.

웹 파일 이외의 동영상이나 사진, 디렉터리 등의 다른 파일을 발견한다면 복사를 수행한다. 웹 파일을 찾은 후, 예외적인 함수를 찾기 위한 1 - PASS 전처리 작업을 수행한 다음, 난독화를 수행한다. 난독화 같은 경우, 로직 변경과 해시 함수 난독화로 구성되어 있다. 난독화를 수행하여 하나의 난독화 된 파일이 만들어진다면, 웹 서버는 난독화 된 파일을 가리켜 웹 서비스를 제공하게 된다. 또다시 난독화를 수행하여 새로운 파일을 생성한다. 이렇게 2 개의 파일을 생성함으로써 웹 서비스는 일정 시간마다 디렉터리 포인트를 변경하고, 변경이 성공적으로 수행된다면 다음 디렉터리의 난독화를 수행한다.

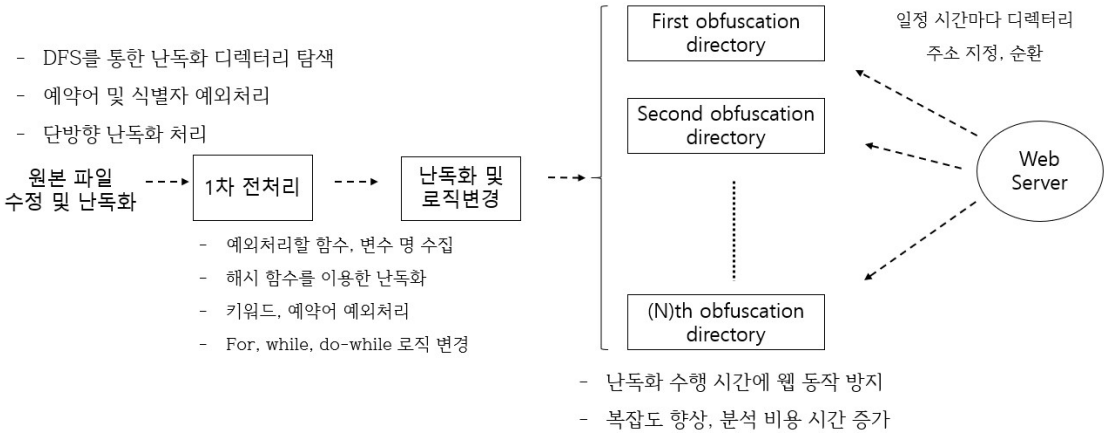

Figure 1. The proposed obfuscation process

예외 키워드 임시 파일은 키워드나 예약어, 난독화를 수행하지 않을 예외적인 함수나 변수 명 등이 입력되어 있는 텍스트 파일을 말한다. ECMA 에서 지정한 키워드, 예약어를 가져왔으며, 1-PASS 전처리를 수행하여 수집되는 예외 함수, 변수 명을 예외 키워드 임시 파일에 저장한다. 그 외 사용자가 단어를 파일에 직접 입력하여 난독화를 수행하지 않을 수 있다.

지정한 디렉터리에서 난독화가 수행될 파일을 찾았을 때, 일차적으로 1-PASS 전처리를 수행한다. 난독화가 수행되지 않을 예외적인 변수와 함수를 찾아서 예외 키워드 임시 파일에 저장하고 2-PASS 난독화를 수행한다. 2-PASS 난독화의 순서는 로직 변경을 먼저 수행한 뒤, 해시 함수로 난독화를 수행 순으로 진행한다.

3.4. 해시 함수 난독화

난독화를 진행하는 방안으로 해시 함수를 이용하였다. 해시 함수는 임의의 길이 데이터를 입력 받아 일정한 길이의 비트열로 변환시켜주는 함수이며 입력 값의 길이가 달라도 출력 값은 항상 고정된 길이로 반환한다. 주로 파일의 Checksum 이나 암호를 저장, 해시 테이블에서 활용을 한다.

기존의 난독화 같은 경우, 특정문자의 난독 결과를 알아내고 다른 값을 유출할 수 있지만 해시 함수와 같은 경우에는 눈사태 효과를 가지기 때문에 입력 값의 아주 작은 변화로도 결과 값이 전혀 다르게 도출되는 효과를 가진다. 예를 들어 입력 값에 점 하나만 추가되어도 기존 값과 전혀 다른 값이 출력되기 때문에 변경되는 부분에 있어 어떠한 규칙성을 찾거나 유추할 수 없다.

해시 함수는 MD5, SHA-1, SHA-256, CRC 등으로 종류가 다양하며 출력된 값의 길이나 저장 공간 등에 따라 사용 용도가 다르다. 해시 함수는 비둘기집의 원리를 가지기 때문에 일정 공간을 넘어가면 다른 입력 값에 같은 출력 값이 나올 수 있는 충돌의 가능성을 가지고 있다. 그러므로 본 연구에서는 충돌이 발생되지 않고 출력 변환 길이가 짧은 SHA-256 해시 함수 알고리즘을 사용한다.

그러나 충돌이 일어날 경우 웹 서비스를 진행하지 못하거나 보안상 큰 문제가 발생할 수 있기 때문에 난독화가 될 대상에 임의의 문자 Salt 값을 사용하는 SHA-256 해시 함수 난독화를

진행한다. Salt 값을 사용함으로써, 해시함수 결과의 충돌가능성을 낮추고, 난독화의 결과값을 정기적으로 바꿈으로써, 유추 가능성 또한 낮출 수 있다.

SHA-512 알고리즘으로 난독화를 수행하는 경우 출력되는 값이 64byte 로 고정되어 있다. 그렇기 때문에 1byte 의 값도 64byte 로 늘어나서 파일의 크기가 매우 커지게 된다. 저장 공간 낭비와 파일을 읽을 때 시간이 증가하는 문제점을 해결하기 위하여 64byte 로 변경된 출력 값을 앞 40byte 만 사용하고 뒤의 24byte 를 떼어내어 난독화를 수행한다. 앞에 임의의 문자가 하나 더 붙으므로 총 41byte 길이가 생성된다. 문자의 길이가 감소함으로써 충돌의 위험성이 커지지만, 랜덤 알파벳 문자 5 자 이상의 Salt 를 적용시켜 충돌이 일어날 가능성을 낮추었다.

만들어진 Salt 는 한 프로그램 내에서만 존재하며 프로그램 종료 시, 소멸이 된다. [Figure 2]의 난독화 된 코드는 다시 수행을 하면 Salt 가 랜덤으로 초기화되기 때문에 난독화 문자가 변경이 된다. 또한 난수의 길이는 사용자가 불규칙하게 정할 수 있으며, 랜덤 Salt 를 붙여 단방향 난독화를 수행하기 때문에 복호화가 불가능하다.

해시 함수를 적용할 때 숫자가 맨 앞에 나올 수 있다. 변수나 함수명은 숫자가 앞에 나올 수 없기 때문에 랜덤 알파벳 문자 하나 이상을 해시 함수가 적용된 소스 앞에 붙여서 난독화를 수행한다.

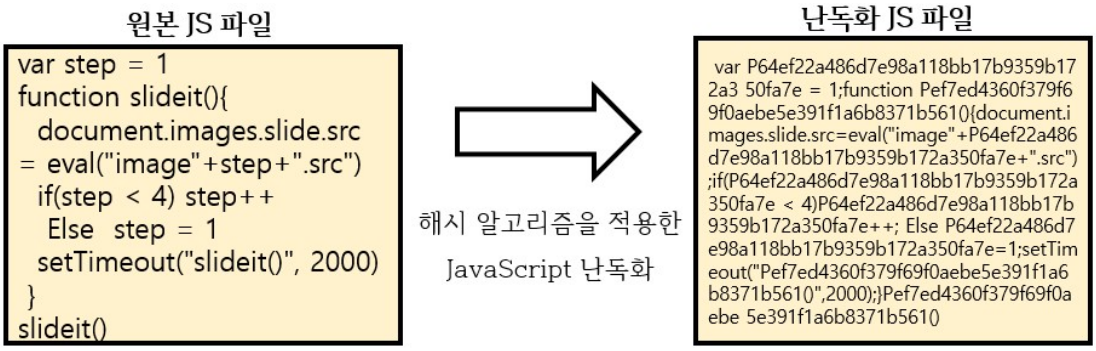

Figure 2. An Example of JavaScript Obfuscation using Hash function

3.5. 로직 변경

해시 함수 난독화와 함께 로직 변경도 함께 진행한다. 변경되는 제어문은 for 문, while 문, do-while 문 등의 3 종이며, 각 제어문의 기능은 그대로 유지한 채로 함수형으로 변경한다[6]. goto 문으로의 변경이 더 단순화된 방식이지만, 자바스크립트가 발전하면서 goto 문의 사용이 매우 제한되기 때문에 본 논문에서는 goto 문을 사용하는 대신 함수형 로직 변경 절차를 제안한다.

로직 변경의 순서는 읽는 과정과 바꾸는 과정의 2 패스로 나누어 진다. 읽는 과정은 우선 일정 길이의 코드를 읽어 평문인지 제어문인지 인식을 하여 첫 번째 배열에 나눠 저장한다. 두 번 배열에는 제어문인 배열과 대응하는 위치에 제어문의 종류를 저장한다. 모든 코드에 대한 읽기 과정이 끝나면, 바꾸는 과정을 실시한다. 바꾸는 과정은 평문과 제어문으로 나눠 놨던 배열 중 제어문을 함수형으로 바꿔주는 과정이다. 두 번째 배열에 저장된 제어문을 확인한 후 그에 대응하는 위치의 첫 번째 배열의 문장을 가져와 제어문 로직 변경을 실행한다. 이와 같은 내용에 대한 사례가 Figure 3 에 도식화되어 설명되어있다.

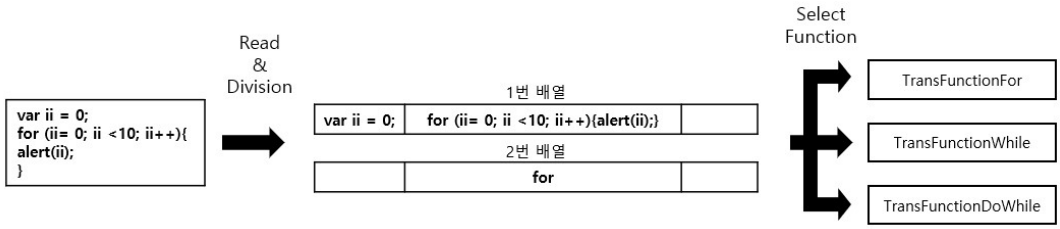

Figure 3. An Example of Logic Change Tasks for a ‘for’ statement

for 문 로직변경을 위해서, 우선 for 문의 선언부, 조건부, 증감, 내용 부분 등으로 4 개의 부분으로 나누어 배열에 저장한다. 지역변수처럼 사용할 수 있도록 중괄호로 변수의 적용 범위(scope)를 만들어준다. 선언부는 Scope 내 최상단에 위치해주며 var 이 붙어 있다면 let 으로 변경한다. 조건부, 증감, 내용 등의 3 개 부분은 함수형으로 만들어준다. 우선 조건부는 for 문에서 if 문이 생략되었기 때문에 함수형으로 전환할 때 if 문을 사용한다. 조건에 부합하는 경우에는, 내용 부분으로 return 하도록 로직을 설정해준다. 내용 부분은 함수형으로 만들어준 후 배열에 있는 내용을 그대로 가져온다. 이후 중복이 되지 않고 if 문 내 break, continue 가 등장하는 경우, break 는 return 0 으로 변경하고, continue 는 return 증감문으로 대체하게 된다. 증감은 함수형으로 선언해 준 후 배열에 저장된 증감 부분을 불러오고 return 조건부를 해준다. 이와 같은 사례가 Figure 4 에 도식화되어 설명되어 있다.

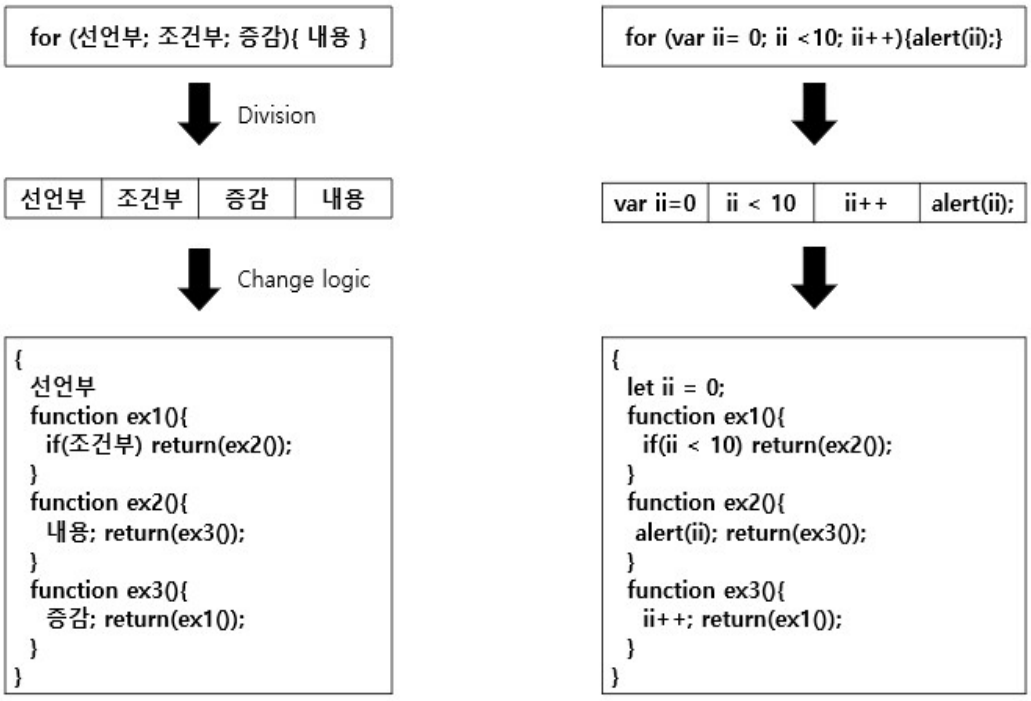

Figure 4. An Example of Logic Change Process for a ‘for’ statement

while 문의 로직 변경의 경우 조건문과 내용을 두 부분으로 나눠 실시한다. 두 부분 모두 함수형으로 변경되며 조건문은 for 문과 마찬가지로 if 문이 생략되었기 때문에 함수형으로 전환할 때 if 문을 사용한다. 조건에 만족하면 return 내용을 실행하도록 로직을 설정해준다. 내용 부분은 함수형으로 만들어준 후 배열에 있는 내용을 그대로 가져온다. 이후 중복이 되지

않고 if 문 내 break, continue 가 등장하는 경우에, break 은 return 0 으로 변경하고, continue 는 return 문으로 대체해준다. 그 외 내용 뒤에 return 문으로 조건문 처리를 해준다.

do-while 은 while 과 로직이 거의 흡사하지만 내용을 먼저 한번 수행한다는 점에서 while 문과 다르다. 그러므로 while 문의 로직 변경에서 내용을 먼저 수행한 후 return 조건을 수행하도록 로직을 설정하면 do-while 의 로직처럼 작동하게 된다.

3.6. 실행 결과

실제 웹 서비스를 실행하기 위해 CentOS 7, Apache 2.4 버전 환경에서 테스트를 진행했다. 난독화 대상 디렉터리의 크기는 이미지와 텍스트 등을 포함해서 총 9.28 MB 이며 HTML 과 자바스크립트 코드는 9609 줄로 이루어져 있다. 예외처리가 될 키워드 및 예약어의 수는 총 198 개이다. 난독화와 로직 변경을 수행한 후 측정한 경과, 약 20(19.553)초와 난독화 되어 크기가 10.2 MB 로 늘어난다. 전체 크기의 11% 포인트 가량 늘어났으며 예외처리가 되는 키워드와 예약어의 수에 따라, 난독화 수행시간에 많은 영향이 끼치는 것을 확인되었다.

3.7. 결과 분석 및 고찰

자바스크립트 난독화 방안에서 인용구에 대한 문자열은 난독화를 않도록 설계하였지만, 간혹 특정 함수에서 인용구 안에서 다른 함수를 호출하는 예외적인 상황이 있다. 예를 들어 setTimeout() 함수 같은 경우, 인용구 안에 호출될 함수 명을 입력해야 하는데 호출되어야 할 함수명은 이미 난독화가 되어 있으므로 인용구 안에 있는 함수와 매칭이 되지 않는 경우가 발생할 수 있다. eval() 함수도 인용구 안에 있는 명령어를 그대로 실행시키기 때문에 eval() 함수 안에서 수행된 명령 중에 변수 명과 함수 명이 있다면 프로그램이 치명적인 문제를 일으킬 수 있다. 그렇다면 인용구 안에 있는 난독화를 수행하거나 일치하는 변수 명, 함수 명을 난독화 시키지 않아야 한다. 이 외에도 여러 가지의 각 함수마다 예외적인 상황을 확인해줘야 하는 문제점이 있다.

또한, 웹 브라우저는 ECMSCRIPT 버전의 업그레이드될 때 마다 사라지는 함수와 변경되는 함수의 구성, 새로운 함수 생성, 키워드, 예약어 등이 발생한다. 그러므로 이에 따른 많은 예외적인 상황을 버전의 업그레이드에 따라 프로그램에서도 수동적으로 업데이트를 해주어야 한다. 만약 업데이트를 수행하지 않는다면, 그것과 관련된 함수를 사용하지 않아도 기존의 함수 구성이 수정이 될 수 있기 때문에 프로그램의 사용이 어려워질 것이다.

제어문을 작성할 때 작성자 습관에 따라 중괄호를 사용하지 않는 경우가 존재한다. 하지만 경우의 수가 너무 다양하므로 중괄호를 사용하지 않는 제어문은 로직 변경을 하지 않는다. 또한, 로직 변경의 경우 return 을 사용하여 스택에 함수들이 쌓이는 현상 때문에 약 15000 개의 함수가 쌓이면 스택 오버플로가 발생한다. 스택 오버플로를 제거하기 위한 주기적인 갱신이 필요하다. 마지막으로 이번 연구에선 단순 문자열 처리로 하였기 때문에 제어문을 작성할 시 빈칸의 여부는 굉장히 유의하여야 한다.

IV. 결론

기술의 발달에 따라 보안의 중요성도 점점 커져가고 있는데 클라이언트 측 보안은 중요시 여겨지지 않으며 보안 위협에 계속적으로 노출이 되고 있다. 본 연구에서는 웹 서비스 클라이언트 측 보안 강화를 위해 난독화 기법을 고안하게 되었다.

구현된 난독화는 해시 함수를 사용하여 단방향으로 난독화를 수행함으로써, 복호화 또는 디코딩을 하는 가능성을 최대한 줄였다. 이로 인해서 소스코드의 복잡도를 증가시켜, 가독성 및 분석 가능성을 매우 낮추었다. 또한 로직 변경을 통해 조건문과 반복문을 함수형으로 변경함으로써 코드 분석의 편이성을 매우 낮추었다.

난독화를 통해 완전한 코드의 보안은 불가능하다. 그러나 난독화를 진행함으로써 코드의 구성을 공격자가 알아보기 어렵고 파악하기 힘들게 하여 분석에 필요한 비용과 시간을 증가시켜 공격을 난해하게 만드는데 의의가 있다.

모바일 시대에 따른 자바스크립트의 수요가 급격하게 증가하고 있다. 본 연구에서 사용된 해시 함수 난독화나 로직 변경 같은 경우는 프로그래밍의 문법에 위반되지 않고 로직과 변수 명, 함수 명을 변경하는 것이기 때문에 웹 서비스뿐만 아니라 자바스크립트가 들어가 있는 다른 프로그램 코드에서도 사용이 가능하다. 또한, 이러한 로직 변경과 난독화 연구 방법을 통하여 자바스크립트뿐만이 아니라 다른 시스템, 서버 등의 프로그래밍 언어에서도 적용이 가능하다.

여러 가지 예외적인 상황과 문제가 있었지만, 향후 이러한 예외적인 상황과 한계점을 해결하기 위해 예외적인 함수에 대한 처리와 외부 호출에 대한 처리를 추가할 것이다. 또한, 문자열 비교 및 정규식을 이용한 실행시간 감소와 저 수준 파일 입출력을 통한 난독화 속도 향상, 그리고 해시 함수 난독화뿐만 아니라, 여러 가지 난독화 기법을 제시하여 강력한 웹 서비스 보안 등의 향후 연구과제이다.

V. Acknowledgments

본 연구는 2018 년도 중소벤처기업부의 기술개발사업 지원에 의한 연구임 [S2658242]

VI. 참고문헌

[1] Young-Kon Kim, Internet Hacking and Security 3rd Ed., Hanbit Publishing Network, 2017.
[2] Young-Ho Kim, Dong-Sik Choi, Sang-Soo Yeo, “A Research of JavaScript Obfuscation Using Random Numbers,” 2019 Summer Conference of ICT Platform Society, pp. 111-113, 2019.
[3] Young-Ho Kim, Dong-Sik Choi, Sang-Soo Yeo, “An Obfuscation Method on JavaScript using Logic Flow Change,” 2019 Summer Conference of ICT Platform Society, pp. 108-110, 2019.
[4] Byeong Yong Lee, Yong Soo Choi,“The Status and Analysis of Obfuscation Techniques and Perspective Development,” (Journal of Security Engineering), Vol.5, No.2, pp. 43-52, 2008.
[5] Hyeyoung Chang, Byoung Min Cho, Seongje Cho, “Comparison of Data Structure and Control Flow Transformations in High-level Obfuscation,” 2007 Congress of the Korean Institute of Information Scientists and Engineers, 34(1D), 89-94, 2007.
[6] Jung-il Kim, Eun-joo Lee, “ A strategy for effectively applying a control flow obfuscation to programs,” Journal of the Korea Society of Computer and Information, Vol. 16, No. 6, pp. 41-50, 2011.