ОБФУСКАЦИЯ КОДА НА ПРИМЕРЕ JAVASCRIPT

Патрушев Даниил Игоревич

студент, Санкт-Петербургский государственный университет телекоммуникаций им. проф. М. А. Бонч-Бруевича, РФ, г. Санкт-Петербург

Костандян Эрик Геворгович

студент, Санкт-Петербургский государственный университет телекоммуникаций им. проф. М. А. Бонч-Бруевича, РФ, г. Санкт-Петербург

Васильев Артём Андреевич

студент, Санкт-Петербургский государственный университет телекоммуникаций им. проф. М. А. Бонч-Бруевича, РФ, г. Санкт-Петербург

Мелкиседекянц Артем Денисович

студент, Санкт-Петербургский государственный университет телекоммуникаций им. проф. М. А. Бонч-Бруевича, РФ, г. Санкт-Петербург

Аннотация. В современном мире мы всё реже встречаемся с распространением программного обеспечения в формах, которые сохраняют большую часть информации, присутствующей в исходном коде. Важным примером является байт-код Java. Поскольку такие коды легко декомпилировать, они увеличивают риск злонамеренных атак с помощью методов деобфускации.

В данной статье мы рассмотрим несколько методов обфусцирования исходного кода в JavaScript. Мы докажем, что автоматическая обфускация кода в настоящее время является наиболее актуальным методом предотвращения деобфускации.

Ключевые слова: Обфускация, ЦВЗ (Цифровой водяной знак), Обратный инжиниринг, Байт-код, минимизатор, Brainfuck

Введение:

Обфускация - это далеко не новый вид защиты программного кода. Просто до недавнего времени этому способу уделялось относительно мало внимания со стороны разработчиков программного обеспечения. Причина заключается в том, что большинство программ являются большими, монолитными и поставляются в виде раздетого собственного кода, что делает их трудными для обратного инжиниринга.

Цель данной статьи - ознакомить читателей с таким понятием, как “обфускация”. Получив физический доступ к приложению, злоумышленник обладая навыками обратного-инжиниринга может декомпилировать его, а затем проанализировать его структуры данных и поток его управления. Это заставляет беспокоиться разработчиков Java. В первую очередь их волнует не столько реинженеринг целых приложений, а сколько перспектива того, что

конкурент сможет извлечь информацию из исходного кода, с целью включить её в собственные программы. В свою очередь такое поведение злоумышленника тяжело обнаружить и преследовать на законных основаниях.

Мы ограничимся обсуждением программ на Java, распространяемых через Интернет в виде файлов классов Java, хотя большинство наших результатов будет применяться и к другим языкам и архитектурам нейтральных форматов. Утверждая, что единственный разумный подход к защите мобильного кода - это запутывание кода.

Классификация обфускации:

Суть процесса защиты программного кода обфускации в том, что трансформированный обфускацией код будет существенно отличаться, но выполнять те же функции, что и изначальный код. Сам процесс в теории выглядит следующим образом:

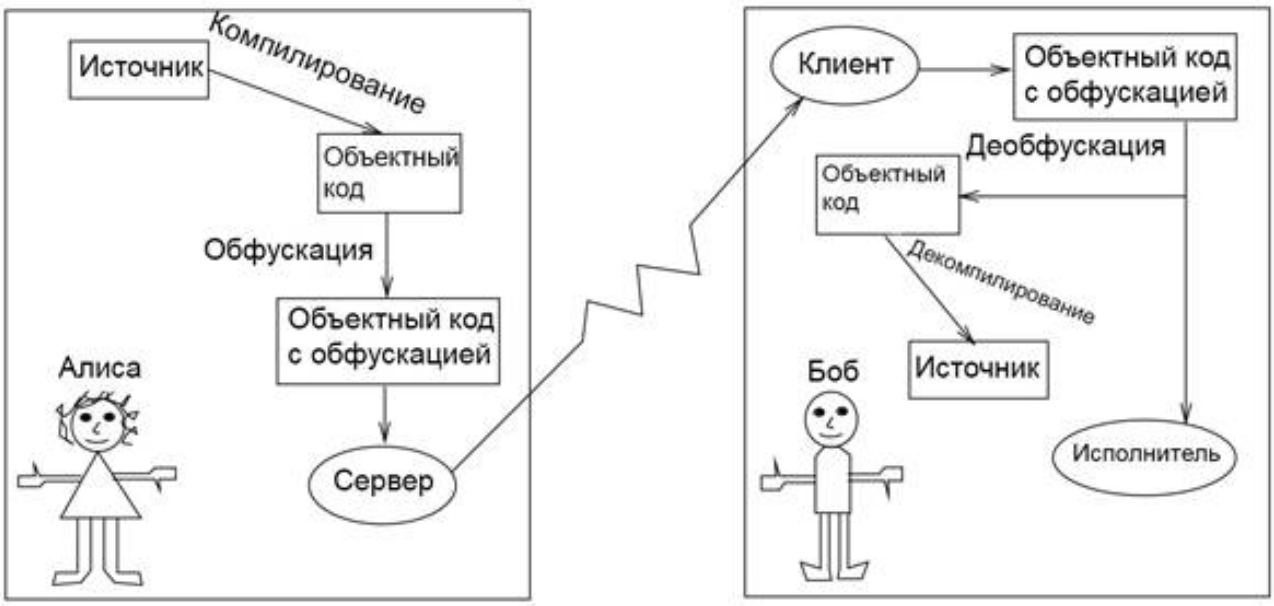

Рисунок 1. Защита Обфускацией

Процессы обфускации можно классифицировать по видам, в зависимости от способа модификации кода программы:

Рисунок 2. Виды Обфусцирования

Оценивать эффективность процесса обфускации принято с помощью аналитического метода, который основываются на трех величинах:

1. Устойчивость - указывает уровень сложности исполнения реверсивной инженерии над кодом
2. Эластичность - указывает эффективность защиты программного кода перед деобфускаторами
3. Стоимость преобразования - позволяет оценить, насколько больше требуется системных ресурсов для выполнения кода прошедшего процесс обфускации, чем для выполнения оригинального кода программы

Методы обфускации в JavaScript:

В таблице снизу мы рассмотрим несколько самых известных и популярных методов обфускации в JavaScript.

Таблица 1.

Методы обфускации

Метод	Суть метода	Устойчивость	Эластичность	Стоимость Преобразования
Обфускация Минимизатором	Удаление пробелов, табуляции и замена значащих имен на переменные(Возможна замена на шестнадцатеричный код)	Низкая	Средняя	Низкая

Обфускация Brainfuck	Метод-головоломка, где всё заменяется на специальные символы	Высокая	Средняя	Высокая
Обфускация невидимостью	Преобразует видимый скрипт в невидимый, то есть в строку состоящую из знаков табуляции (бит 1) и пробелов (бит 0)	Средняя	Низкая	Высокая
Обфускация флудом	В скрипт, а также в комментарии вставляется флуд, не несущий смысловой нагрузки	Средняя	Низкая	Средняя
Обфускация несуществующими функциями	Обфусцируем код не на уровне шифровки строк, а на уровне получения строк от самого JS.	Средняя	Высокая	Низкая

Заключение:

Обфускация методом Brainfuck является одной из самых трудных для реверс-инжиниринга, а обфускация минимизаторами является простым, для реверс-инжиниринга.

Метод обфускации несуществующими функциями предоставляет самую высокую защиту перед деобфускаторами, в отличие от таких методов как: обфускация флудом и невидимостью.

Таким образом, в двойку наших лидеров входят методы: обфускации Brainfuck и несуществующими функциями. Но стоит учесть, что обфускация Brainfuck ресурсозатратнее. Идеальной альтернативой в данном случае является - связка обоих методов.

Список литературы:

1. Ручная деобфускация JavaScript - [Электронный ресурс] - https://www.securitylab.ru/analytics/424404.php

2. Таксономия Обфускационных Трансформаций - [Электронный ресурс] - http://www.reverse-engineering.info/OBF/A4.pdf //Christian Colberg, Clark Thomborson, Douglas Low

3. Обфускация JavaScript - [Электронный ресурс]- https://habr.com/ru/post/112530/

4. Обфускация и защита программных продуктов - [Электронный ресурс]- http://citforum.ru/security/articles/obfus/

5. Красов А.В. Методика защиты байт-кода java-программы от декомпиляции и хищения исходного кода злоумышленником / Красов А.В., Шариков П.И. // Вестник Санкт-Петербургского государственного университета технологии и

дизайна. Серия 1: Естественные и технические науки. 2017. № 1. С. 47-50.

6. Комилджонов Р.Н. Обфускация кода на примере JavaScript / Комилджонов Р.Н., Косов Н.А. // Colloquium-journal. 2019. № 13-2 (37). С. 71-75.