International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

COPY RIGHT

IJIEMR Transactions, online available on 22nd Mar 2022. Link :http://www.ijiemr.org/downloads.php?vol=Volume-11&issue=ISSUE-02

DOI: 10.48047/IJIEMR/V11/I03/13

Title **Client-side cross-site scripting protection methods**

Volume 11, Issue 03, Pages: 71-75

Paper Authors
Kerimov Kamil Fikratovich, Tolipov Dilshod Ali ogli

USE THIS BARCODE TO ACCESS YOUR ONLINE PAPER

To Secure Your Paper As Per UGC Guidelines We Are Providing A Electronic Bar Code

International Journal for Innovative Engineering and Management Research
A Peer Reviewed Open Access International Journal
www.ijiemr.org

УДК 004.056.53

Методы защиты отмежсайтовогоскриптингана стороне клиента

Керимов КамилФикратович
Phd, заведующий кафедрой систем и прикладного программирования
Тел: +998 90 909 70 01
Эл.почта:kamil@kerimov.uz
ТолиповДилшод Али угли
Магистрант ташкентского университета информационных технологий
Тел: +998 97 704 66 06
Эл.почта: dee.talipoff@gmail.com

В последние годы киберпреступники нацелились на веб-сайты и социальные сети, чтобы внедрить вредоносные скрипты, которые ставят под угрозу безопасность посетителей таких веб-сайтов. Это включает в себя выполнение действий с использованием браузера жертвы без его разрешения. Это создает потребность в разработке эффективных механизмов защиты от веб-атак, которые в основном нацелены на конечного пользователя. В этой статье решается вышеуказанные проблемы с точки зрения предотвращения исполнения вредоносного кода JavaScript на стороне клиента. Предлагаемые модели фильтрует входящие данные и устанавливает флаги безопасности контента, которые в контексте веб-атак на стороне клиента обеспечит более безопасную среду просмотра для конечного пользователя.
Ключевые слова:Атака на стороне клиента, XSS, AJAX, Вредоносный JavaScript.

Керимов К.Ф., Толипов Д.А.
Сўнги йилларда кибер жиноятчилар веб-сайтларга ва ижтимоий тармоқларга зарарли скриптларни юклашни мақсад қилиб олишган, бу ўз навбатида фойдаланувчиларнинг хавфсизлигини хавф остига қўяди.Шу билан бирга жабрланувчиларнинг рухсатисиз браузеридан фойдаланиш холатларини ўз ичига олади. Бу биринчи навбатда фойдаланувчиларни самарали веб-хужумлардан ҳимоя механизмларини ишлаб чиқиш заруратини келтириб чиқармоқда. Ушбу мақола фойдаланувчи томонида зарарли JavaScript кодини бажарилишини олдини олиш нуқтаи назаридан юкоридаги масалаларни кўриб чиқади. Таклиф этилаётган модель кирувчи маълумотларни фильтрлайди хамда фойдаланувчи томонидаги веб-хужумлар контекстида хавфсизроқ кўриш тажрибасини таъминлайдиган контент хавфсизлиги байрокларини ўрнатади.
Калит сўзлар:Фойдаларувчи томонидаги хужум, XSS, AJAX, Зарарли JavaScript

International Journal for Innovative Engineering and Management Research
A Peer Reviewed Open Access International Journal
www.ijiemr.org

Client-side cross-site scripting protection methods

Kerimov Kamil Fikratovich
Phd, Head of the department of systems and applied programming in Tashkent University of Information Technologies named after Al-Khwarizmi
Phone: +998 90 909 70 01
Email: kamil@kerimov.uz
Tolipov Dilshod Ali ogli
Master of Tashkent University of Information Technologies named after Al-Khwarizmi
Phone: +998 97 704 66 06
Email: dee.talipoff@gmail.com

In recent years, cybercriminals have targeted websites and social media to inject malicious scripts that compromise the security of visitors to such websites. This includes performing actions using the victim's browser without their permission. This creates a need to develop effective web attack protection mechanisms that are primarily aimed at the end user.This article addresses the above issues in terms of preventing the execution of malicious JavaScript code on the client side. The proposed model filters incoming data and sets content security flags that, in the context of client-side web attacks, will provide a more secure browsing experience for the end user.

Keywords: Client-side attacks, XSS, AJAX, Malicious JavaScript

I. ВВЕДЕНИЕ

Использование веб-браузера для просмотра веб-страниц подвергает конечного пользователя постоянно растущему числу атак. Эти виды атак варьируются от фишинговых атак (сбор информации) до реальных эксплойтов в веб-браузере, которые могут привести к полной компрометации компьютера конечного пользователя, чтобы он стал частью ботнета. Тем не менее, настоящая проблема заключается не только в защите конечного пользователя при просмотре веб-страниц, но и в сохранении удобства для пользователя. В хорошо защищенных средах, чтобы позволить веб-браузеру подключаться к Интернету, брандмауэры должны быть настроены на открытие определенных номеров портов (например, порт 80 дляHTTP и 443 для HTTPS). Операционная система (ОС), IDS на основе хоста и антивирусные системы должны предоставить разрешения веб-браузеру для выполнения плагинов, связи с внешним миром и доступа к файловой системе в случае,

если конечный пользователь хочет загрузить / загрузить некоторые файлы на / со своей машины. Поистине иронично видеть, сколько времени и усилий специалисты по безопасности тратят на создание многоуровневых средств защиты, которые настраиваются таким образом, чтобы позволить веб-браузеру свободно просматривать веб-страницы. В противном случае конечные пользователи будут жаловаться, что их опыт просмотра очень ограничен.

Веб-приложение на основе Web 2.0 и AJAX еще больше решает эту проблему. Веб-сайт, созданный с использованием технологий AJAX, может содержать код, который устанавливает обратное соединение с сервером. Такое соединение позволит обойти всевозможные многослойные защиты в основном по двум причинам. Первая причина заключается в том, что защита настроена так, чтобы разрешать такие соединения, поскольку большинство популярных веб-сайтов используют AJAX. Вторая причина заключается в том, что трафик может быть зашифрован или сильно запутан, что сделает всю сетевую защиту полностью слепой к тому, что передается обратно на сервер. Большинство атак, использующих такие возможности связи, подпадают под действие так называемых атак межсайтового скриптинга (XSS). Семейство XSS включает несколько методов атаки, включая канал XSS, туннель XSS, червь XSS и подделку межсайтовых запросов (CSRF). XSS - это своего рода инъекционная атака, которая в основном обходит ту же политику происхождения, что и веб-браузеры. Политика одинакового происхождения позволяет сценариям, созданным с одной и той же веб-страницы, получать доступ к конфиденциальной информации в клиентском браузере, такой как файлы cookie HTTP для аутентификации сеанса. Если злоумышленник сможет внедрить вредоносный сценарий на легитимную веб-страницу, которая выполняет XSS-атаку, он сможет получить конфиденциальную информацию при каждом посещении пользователем такой зараженной веб-страницы. Такая же политика происхождения также применяется к XMLHttpRequests, который является частью AJAX.

В этой статье предлагается подход к смягчению атак на стороне клиента путем фильтрации пользовательского ввода в JavaScriptи политика безопасности контента(CSP).

II. МЕТОДЫ ЗАЩИТЫ ОТ ВЕБ-АТАК НА СТОРОНЕ КЛИЕНТА

Веб-страницы состоят из HTML, обычно описанного в файлах шаблонов, с динамическим содержимым, вплетенным при рендеринге страницы. XSS-атаки используют неправильную обработку динамического контента, поступающего из серверного хранилища данных или поисковых запросов клиента. Злоумышленник злоупотребляет редактируемым полем для вставки кода JavaScript, и он исполняется при загрузке страницы.

International Journal for Innovative Engineering and Management Research
A Peer Reviewed Open Access International Journal
www.ijiemr.org

Во избежание необработанного HTMLнеобходимо экранировать все динамическое содержимое, поступаемое из хранилища данных, чтобы браузер знал, что его следует рассматривать как содержимое HTML-тегов.

Экранирование динамического содержимого обычно состоит из замены значащих символов кодировкой HTML-сущности, например:

- " &#34
- # &#35
- & &#38

Чтобы избежать пользовательского ввода в контексте HTML в JavaScript, необходим собственный HTML-кодер, потому что JavaScript не предоставляет API для кодирования HTML. Вот несколько примеров кода JavaScript, который преобразует строку в HTML-сущности:

```
function htmlEncode(str){
    return String(str).replace(/[^\w. ]/gi, function(c){
        return '&#'+c.charCodeAt(0)+';';
    });
}
```

Затем эту функцию можно использовать следующим образом:

```
<script>document.body.innerHTML = htmlEncode(untrustedValue)</script>
```

Если ввод находится внутри строки JavaScript, то нужен кодировщик, который выполняет экранирование Unicode. ВотпримерUnicode-кодера:

```
function jsEscape(str){
    return String(str).replace(/[^\w. ]/gi, function(c){
        return '\\u'+('0000'+c.charCodeAt(0).toString(16)).slice(-4);
    });
}
```

Последней линией защиты от межсайтовых сценариевявляется политика безопасности контента (CSP). Если предотвращение XSS не удается, для смягчения XSS используется CSP, ограничив то, что может сделать злоумышленник.

CSP позволяет управлять различными вещами, например, можно ли загружать внешние скрипты и будут ли выполняться встроенные сценарии. Для развертывания CSP необходимо включить заголовок ответа HTTP под названием Content-Security-Policy со значением, содержащим политику.Пример CSP выглядит следующим образом:

```
default-src 'self'; script-src 'self'; object-src 'none'; frame-src 'none'; base-uri 'none';
```

Эта политика определяет, что такие ресурсы, как изображения и сценарии, могут быть загружены только из того же источника, что и главная страница. Таким образом, даже если злоумышленник может успешно ввести полезную нагрузку XSS, он может загружать ресурсы только из текущего источника. Это значительно снижает вероятность того, что злоумышленник сможет использовать уязвимость XSS.

International Journal for Innovative Engineering and Management Research
A Peer Reviewed Open Access International Journal
www.ijiemr.org

Если требуется загрузка внешних ресурсов, прежде всего необходимо убедится, что разрешается использовать сайту только те скрипты, которые не помогают злоумышленнику. Например, если внести определенные домены в белый список, злоумышленник может загрузить любой сценарий из этих доменов.

Если это невозможно, вы можете использовать политику на основе хэша или nonce, чтобы разрешить сценарии в разных доменах. Nonce— это случайная строка, которая добавляется в качестве атрибута скрипта или ресурса, который будет выполняться только в том случае, если случайная строка совпадает с сервером. Злоумышленник не может угадать рандомизированную строку и, следовательно, не может вызвать сценарий или ресурс с допустимым nonce, поэтому ресурс не будет выполнен.

Предлагаемые подходы позволятьблокировать скрытые веб-атаки, которые крадут информацию о конечных пользователях без их согласия. Атаки, как XSS, труднее всего обнаружить, поскольку они происходят внутри веб-браузера. Все эти атаки могут быть выполнены с использованием необфусцированногоJavaScript, который использует стандартные веб-технологии, такие как AJAX, HTML и CSS. Одни и те же атаки могут быть выполнены разными способами.

III. ЗАКЛЮЧЕНИЕ

В этой статье были рассмотреныметодыпредотвращения или смягчения атак XSSна стороне клиента, которыеминимизируют утечку информации.

Предлагаемыеметодыобеспечивают структуру, которые при применении в настройках веб-браузера ограничивает загрузку исполняемогоJavaScriptкода из неизвестных источников, фильтрации необработанных HTMLв динамических веб сайтах и экранирования символов сущности HTML.

ИСПОЛЬЗОВАННАЯ ЛИТЕРАТУРА

1. Rustam Kh. Khamdamov, Komil F. Kerimov, JalolOybekugli Ibrahimov, Method of Developing a Web-Application Firewall // Journal of Automation and Information Sciences DOI: 10.1615/JAutomatInfScien.v51.i6.60 New York, USA 6,2019

2. Рябко Д.М. Подход к тестированию уязвимости web-приложений от атак типа SQL-инъекций // УкрПРОГ, Киев, Украина, 2006.

3. Ю.Ю. Громов, В.О. Драчев, О.Г. Иванова. Информационная безопасность и защита информации: Учебное пособие: — Ст. Оскол: ТНТ, 2017. — 384 с.

4. Низамутдинов М.К. Тактика защиты и нападения на ИТ-приложения. – Санкт-Петербург: БХВ-Петербург, 2005. – С. 30 - 60.

5. Запечников, С.В. Информационная безопасность открытых систем. В 2-х т. Т.1 — Угрозы, уязвимости, атаки и подходы к защите М.: ГЛТ, 2017. — 536 с.