НАУКОВО-ТЕХНІЧНИЙ ЖУРНАЛ «ЗАХИСТ ІНФОРМАЦІЇ» №1, 2011

УДК 004.056.53 **Коломыцев М.В., Носок С.А. (НТУУ «КПИ»)**

СЕРВИС НЕОТКАЗУЕМОСТИ В WEB-ПРИЛОЖЕНИЯХ

Задача безопасной обработки данных в сетевой среде требует обеспечения подотчетности действий пользователей, участвующих в обработке. Традиционные средства регистрации и аудита в распределенных системах оказываются неэффективными. В таких ситуациях, для отслеживания действий пользователя можно использовать сервис неотказуемости. В статье, на примере Web-приложения, анализируются возможные уязвимости процесса обмена информацией, и предлагается подход к обеспечению неотказуемости источника.

Неотказуемость (англ. *non-repudiation*) — невозможность отказа от авторства. Приложение обеспечивает сервис неотказуемости, если оно гарантировано устанавливает соответствие между пользователем (или процессом, действующем от имени пользователя) и данными, которые пользователь создает, модифицирует либо пересылает с помощью приложения. Такими данными может быть файл, сообщение, аутентификационная информация пользователя, либо конфигурационная информация самого приложения.
Наиболее распространенными и часто обсуждаемыми вариантами неотказуемости являются неотказуемость источника (когда пользователь не может ложно отказаться от отправки сообщения или документа) и неотказуемость получения (когда пользователь не может ложно отказаться от получения сообщения или документа). Однако могут быть определены и другие варианты, включая неотказуемость создания, неотказуемость доставки и неотказуемость подтверждения.

Существует обширный класс приложений, для которых сервис неотказуемости является обязательный (например, финансовые транзакции). Кроме того, сервис неотказуемости важен для приложений, в которых необходимо обеспечивать подотчетность действий пользователя для каждой совершенной транзакции, в частности, если эти транзакции совершаются через сетевую среду с использованием различных компонент приложения. Например, для почтовых сообщений такими компонентами являются почтовый сервер и почтовый клиент, для Web это браузер и Web-сервер. В таких случаях традиционных средств регистрации и аудита недостаточно для поддержания свойства подотчетности. Задача отслеживания действий пользователя в таких случаях может решаться с использование сервиса неотказуемости.

Основная идея невозможности отказа от определенного действия состоит в том, что пользователь криптографически связан со специфическим действием таким образом, что последующий отказ от этого действия равнозначен в некоторой степени признанию злого умысла или небрежности пользователя [1]. Сервис неотказуемости может базироваться только на инфраструктуре открытых ключей, но не на инфраструктуре симметричных ключей. Для связи в среде, где используются симметричные ключи, пользователи должны создавать общий симметричный ключ, то есть разделять общий секрет, что дает возможность одному из пользователей подделывать подтверждения подлинности.
В качестве примера, рассмотрим случай, когда сервис неотказуемости необходим в Web-приложениях.

Подмена параметров в Web-приложениях

Подмена параметров это класс атак, в котором атакующий подменяет данные, пересылаемые клиентским приложением (браузером) Web-серверу [2,3]. В большинстве случаев, для организации ввода информации пользователем Web-приложения используют HTML-формы. Проблема безопасности HTML-форм заключается в том, что по своей природе Web-приложения наследуют свойства HTTP соединений – для каждой новой формы устанавливается новое соединение, не сохраняющее свойств предыдущего соединения. В

этом смысле Web-приложения являются приложениями «без предыстории» (stateless). Если приложение взаимодействует с пользователем с помощью набора форм для ввода информации, то возникает необходимость сохранения некоторых данных, введенных на предыдущих шагах, чтобы передать собранную информацию на сервер как часть следующей транзакции. Чтобы каким-то образом сохранить контекст предыдущих форм, разработчики могут использовать два подхода – сохранять необходимую информацию на сервере либо на стороне клиента, причем большинство разработчиков предпочитают второй вариант, считая его более легким в реализации. Информация на стороне клиента может сохраняться в файлах *cookies* браузера, в виде значений, добавляемых к URL и в скрытых полях HTML-форм.

К сожалению, каждый из этих вариантов уязвим к атакам. Главное правило, каким должен руководствоваться разработчик гласит: любая информация, возвращаемая клиентом на сервер может быть подменена – значения полей форм, HTTP-заголовки и даже *cookies*.

Подмена информации в скрытых полях

Наиболее общим подходом к хранению информации является использование скрытых полей, т.е. полей, которые встроены в форму, но которые браузер не отображает. В исходном коде скрытые поля имеют вид, аналогичный следующему:

```
<input type="hidden" name="userid" value="ktrout">
<input type="hidden" name="form_expires"
value="20001001:12:45:20">
```

Однако, HTML-форма может быть сохранена пользователем на клиентском компьютере, а значения «скрытых» полей могут быть прочитаны и изменены. Затем HTML-форма загружается в браузер и, если приложение доверяет содержимому формы, вместе с измененными значениями передается на сервер. Для защиты от таких атак приложение может анализировать значение заголовка HTTP_REFERER, которое формируется браузером. В случае сохранения формы на диске и последующей его загрузки, значение HTTP_REFERER будет пустым. Однако такая защита может быть преодолена, поскольку замена заголовка может быть выполнена с помощью несложного скрипта (например, на Perl).

Подмена информации в URL

Если для обмена информацией между страницами (фактически, между сценариями, генерирующими страницы), используется URL-адрес новой страницы, то в строку URL после знака **?** добавляется значение переменной в виде пары *имя_переменной=значение*. Если необходимо передать несколько значений, то пары *имя_переменной=значение* разделяются знаком **&**. С точки зрения безопасности метод имеет очевидный недостаток – значение URL отображается в адресной строке браузера и легко может быть изменено.

Так, если Web-приложения в качестве хранилища данных использует базу данных, то URL может содержать SQL-запрос к базе. Атакующий может манипулировать кодом SQL для доступа к конфиденциальной информации сервера.

Подмена информации в *cookie*

В случае, если информация о текущей сессии пользователя хранится на сервере (в виде временных файлов или временных таблиц в базе данных), ссылки на эту информацию должны быть доступны на стороне клиента. Как правило, они хранятся в виде ***Id*** сессии в файлах *cookie*. *Cookie* могут использоваться сервером для опознания ранее аутентифицированных пользователей. В этом случае, сервер, по окончании процедуры аутентификации, отправляет страницу успешного входа, прикрепив *cookie* с неким

идентификатором сессии. Эта *cookie* может быть действительна только для текущей сессии браузера, но может быть изменена и на длительное хранение. Каждый раз, когда пользователь запрашивает страницу с сервера, браузер автоматически отправляет *cookie* с идентификатором сессии серверу. Сервер проверяет идентификатор по своей базе идентификаторов и, при наличии в базе такого идентификатора, разрешает пользователю доступ.

Файлы *cookie* - это текстовые файлы, содержащие пары имя/значение. Кроме того, они могут содержать информацию о сроке действия, пути и доменном имени (RFC2965). Домен и путь говорят браузеру, что *cookie* должна быть отправлена обратно на сервер при запросах URL для указанного домена и пути. Если они не указаны, используются домен и путь запрошенной страницы. Дата истечения указывает браузеру, когда удалить *cookie* . Если срок истечения не указан, *cookie* удаляется по окончании пользовательского сеанса, то есть с закрытием браузера. Это так называемые сессионные *cookie*. Если же указана дата истечения срока хранения, *cookie* становится постоянной до указанной даты. Браузер MS Internet Explorer хранит их в виде отдельных файлов в папке *Application Data\cookies*. Браузеры на основе Netscape и Mozilla хранят их в одном файле, называемом *cookies.txt*. А поскольку они сохраняются в файл, то могут быть изменены с помощью текстового редактора.

Сессионные *cookie* подделать сложнее, однако существуют программные средства, например, *Burp Proxy* [4] , которые позволяют просматривать и изменять сессионные *cookie* .

Обнаружение подмены параметров с помощью цифровой подписи

Универсальным методом, позволяющим защитить информацию от подмены, является цифровая подпись (ЭЦП). С помощью ЭЦП можно обеспечить неотказуемость на любом этапе обмена информацией между клиентом и сервером. Для этого приложение должно обеспечивать выполнение следующих функций:

- **Подтверждение при передаче**. Приложение должно поддерживать технологию цифровой подписи (например, SHA-1, DSA, RSA), позволяющую отправителю (получателю) заверить цифровой подписью данные или ключи, которые приложение использует при создании или пересылки данных через сеть.
- **Подтверждение доставки**. В этом случае получатель должен с помощью цифровой подписи через приложение подтвердить факт получения информации.
- **Проверка цифровой подписи**. В приложении должна быть реализована возможность проверки соответствия данных (или ключей) полученных по сети или из баз данных, и цифровой подписи, прикрепленной к этим данным.
- **Защита информации, используемой для формировании цифровой подписи**. Приложение должно надежно защищать данные используемые при формировании цифровой подписи от подмены или раскрытия. К таким данным относятся ключи шифрования и сертификаты.
- **Цифровая подпись данных, полученных из Web-форм**. Чтобы сделать невозможным отказ от авторства для конфиденциальных данных, полученных из Web-форм, приложение должно подтверждать их цифровой подписью.

Для защиты данных в HTML-формах можно использовать алгоритм MD5 (либо другой алгоритм) для формирования цифровой подписи под скрытыми полями. Можно, например, объединить значения скрытых полей путем их конкатенации, создать для этой строки цифровую подпись и записать ее в еще одно скрытое поле. Когда браузер возвратит на сервер значения полей формы, их целостность может быть проконтролирована.

Для того, чтобы пользователь не мог подделать цифровую подпись, необходимо добавить к ЭЦП некий секретный компонент, который пользователь не может видеть. Это делается с помощью секретного ключа, хранимого на сервере. Такая конструкция называется "message

authentication code" (MAC), а стандартный способ ее применения называется HMAC (RFC 2104). Функции HMAC доступны во всех языках Web-программирования:

- Digest::HMAC (Perl, [5])
- mhash() (PHP, [6])
- KeyGenerator (Java, [7])
- HMACSHA1 (MS .Net Framework)

Такой подход так же может быть использован для Web-приложений, хранящих данные на сервере и использующих *cookie* для хранения на стороне клиента сессионного ключа.

Конечно, уязвимостью данного метода (как и других методов, основанных на «разделяемых» ключах), является секретный ключ, хранимый на Web-сервере. Если сервер взломан, либо становится доступным исходный код приложения, данный метод перестает быть безопасным. Если же существует обоснованная уверенность в безопасности сервера, и при регулярной смене секретного ключа, можно считать что метод обеспечивает достаточную защиту.

Література

1. Gary McGraw, Software Security: Building Security In. Addison Wesley Professional. Pp 448, 2006, Addison Wesley Professional. ISBN-10: 0-321-35670-5
2. Herbert H. Thompson and Scott G. Chase. The Software Vulnerability Guide. CHARLES RIVER MEDIA, INC. I SBN 1-58450-358-0, 2005, pp 354
3. Самоучитель хакера : подроб. иллюстрир. рук.: [учеб, пособие] / Alex Atsctoy. — М.: Лучшие книги, 2005. — 192 с.: ил. — ISBN 5-93673-036-0. УДК 004.056.53(075.8)
4. http://portswigger.net/proxy/
5. http://search.cpan.org/
6. http://mhash.sourceforge.net/
7. http://java.sun.com/products/jce/

Надійшла: 13.03.11

Рецензент: д.т.н., проф. Литвиненко О.Є.