ректность всех доверенных субъект-сессий относительно всех доверенных субъект-сессий и сущностей);

— для всех субъект-сессий $s \in S_N$ выполняются равенства $\{s' \in S_N : f_{s_N}(s') = f_{s_N}(s)\} \times (E_N \cup S_N) \subset af\_correct_N(s) = ap\_correct_N(s)$ (абсолютная функциональная и параметрическая корректность субъект-сессии относительно всех сущностей и субъект-сессий с совпадающим уровнем конфиденциальности).

Тогда система $\Sigma(G^*, OP, G_0)$ безопасна в смыслах Белла — ЛаПадулы и мандатного контроля целостности.

Условия теоремы БТБ-ДП требуют от ОССН функционально и параметрически корректной (абсолютно корректной) реализации всех субъект-сессий и корректного задания соответствующих уровней конфиденциальности и целостности функционально или параметрически ассоциированных с ними сущностей. Если, например, субъект-сессия, имеющая высокий уровень доступа, некорректно обрабатывает данные («заражается») в сущностях с низким уровнем конфиденциальности и это приводит к получению фактического владения над нею субъект-сессией с низким уровнем доступа, то система защиты ОССН не сможет этому воспрепятствовать. Таким образом, условия теоремы БТБ-ДП указывают на необходимость повышения качества разработки прикладного программного обеспечения ОССН.

ЛИТЕРАТУРА

1. *Девянин П. Н.* Модели безопасности компьютерных систем. Управление доступом и информационными потоками: учеб. пособие для вузов. 2-е изд., испр. и доп. М.: Горячая линия — Телеком, 2013. 338 с.
2. *Девянин П. Н.* Адаптация мандатной сущностно-ролевой ДП-модели к условиям функционирования ОС семейства Linux // Системы высокой доступности. 2013. № 3. С. 98–102.
3. *Девянин П. Н.* Администрирование системы в рамках мандатной сущностно-ролевой ДП-модели управления доступом и информационными потоками в ОС семейства Linux // Прикладная дискретная математика. 2013. № 4(22). С. 22–40.
4. Операционные системы Astra Linux. `http://www.astra-linux.ru/`.

УДК 004.94

ОБЩИЙ МЕТОД АУТЕНТИФИКАЦИИ HTTP-СООБЩЕНИЙ В ВЕБ-ПРИЛОЖЕНИЯХ НА ОСНОВЕ ХЕШ-ФУНКЦИЙ

Д. Н. Колегов

Предлагается метод аутентификации HTTP-сообщений в веб-приложениях, построенный на основе криптографических протоколов с ключевыми хеш-функциями. Данный метод может быть использован для защиты от многих атак на веб-приложения, использующих уязвимости в реализации механизмов аутентификации или авторизации.

Ключевые слова: *криптографические протоколы, аутентификация сообщений, веб-приложения.*

Одним из свойств, характеризующих безопасность протоколов, является свойство аутентификации сообщений, заключающееся в обеспечении аутентификации источника данных и целостности передаваемого сообщения. Аутентификация источника данных означает, что протокол обеспечивает гарантии того, что полученное сообщение или

его части были созданы одним из участников протокола в некоторый момент времени, предшествующий получению сообщения, и что никакие данные не были модифицированы или подделаны. Считается, что аутентификация источника данных включает целостность данных [1].

В веб-приложениях аутентификация сообщений, которыми обмениваются клиент и сервер по протоколу HTTP, как правило, реализуется на уровне логики самого веб-приложения и позволяет обеспечить выполнение следующих свойств безопасности:

— аутентичность источника HTTP-запроса;
— целостность имен и значений параметров, переданных клиенту в HTTP-ответе в виде HTML;
— целостность потока операций.

Кроме того, аутентификация сообщений на основе криптографических методов защиты позволяет существенно повысить уровень защищённости веб-приложения, а также его стойкость к средствам автоматизированного анализа [2]. У известных механизмов аутентификации сообщений в веб-приложениях [3 – 7] можно выделить следующие недостатки:

— отсутствует общая формальная модель безопасности механизма аутентификации сообщений;
— разработанные методы, как правило, ориентированы на конкретное веб-приложение и часто не учитывают особенности криптографических протоколов (например, необходимость защиты от атак повтора или от утечек CSRF-токенов, необходимость использования уникальных токенов);
— отсутствуют механизмы контроля целостности и валидации данных, генерируемых на клиентской стороне (например, данных, вводимых в поля форм).

Аутентификация HTTP-сообщений рассматривается как элемент политики атрибутного управления доступом. На основе положений модели ABAC [8] строится общая модель аутентификации сообщений.

Используются следующие обозначения: A — алфавит; A^* — множество всех слов в A; $[a]$ — множество букв слова a; $x||y$ — конкатенация строк x и y; h — ключевая хеш-функция, построенная по алгоритму HMAC.

Построим модель аутентификации HTTP-сообщений на основе модели ABAC. Основными элементами модели являются следующие: O — множество объектов; S — множество субъект-сессий; $E = S \cup O$ — множество сущностей; OA — множество атрибутов объектов; SA — множество атрибутов субъект-сессий; $EA = ES \cup EO$ — множество атрибутов сущностей; OP — множество операций (видов доступа); TA — множество всех типов атрибутов; SAV — множество значений атрибутов субъект-сессий; OAV — множество значений атрибутов объектов.

В рамках модели ABAC будем считать HTTP-запросы пользовательскими субъект-сессиями; параметры и заголовки HTTP-запроса, идентификатор субъект-сессии (пользователя) — атрибутами субъект-сессии; элементы (scheme, authority, path) схемы URI HTTP-ресурсов — объектами; разрешённые параметры запроса, разрешённые идентификаторы пользователей, запрашивающих доступ к ресурсу, — атрибутами объекта; методы протокола HTTP — видами доступов. Будем также считать, что субъект-сессия имеет обязательные атрибуты key и $time$, а каждый объект имеет обязательный атрибут mac.

Определим функции:

— $type : E \to TA$ — функция, задающая тип атрибута;

— $EEA : E \to 2^{OA} \cup 2^{SA}$ — функция, задающая множество атрибутов сущности;
— $EAT : E \times T \to 2^{OA} \cup 2^{SA}$ — функция, задающая для сущности множество атрибутов данного типа;
— $AV : E \times EA \to SAV \cup OAV$ — функция, определяющая значение атрибута сущности;
— $assign\_auth : O \times 2^{EA} \times OP \to SAV \cup OAV$ — функция, вычисляющая значение переменной de_jure_auth для соответствующей сущности-объекта и значение переменной de_facto_auth для сущности субъект-сессии.

Пусть заданы множества S, O, OA, SA, TA, OP, EC, SAV, OAV и функции EEA, EAT, $type$, AV и $assign\_auth$. Определим предикат $can\_access(s, o, op)$, истинный тогда и только тогда, когда выполнены следующие условия:

1) для объекта o определен атрибут mac со значением $AV(o, mac) = h(AV(s, key), de\_jure\_auth, AV(s, time))$, где $de\_jure\_auth = assign\_auth(o, 2^{EEA(o)}, op)$;
2) выполнено равенство $AV(o, mac) = h(AV(s, key), de\_facto\_auth, AV(s, time))$, где $de\_facto\_auth = assign\_auth(o, 2^{EEA(s)}, op)$.

Назовём $P = (can\_access(s, o, op), assign\_auth)$ политикой безопасности. Будем говорить, что HTTP-запрос $s \in S$ на доступ $op \in OP$ к ресурсу $o \in O$ является аутентичным, если предикат $can\_access(s, o, op)$ является истинным.

Определим политику безопасности P так, чтобы стало возможным обеспечить выполнение одного или нескольких свойств безопасности веб-приложений.

Параметр (атрибут) $q \in OA$ объекта $o \in O$ называется контролируемым по значению, если политика безопасности обеспечивает выполнение условия $AV(o, q) = = AV(s, q)$.

Параметр $q \in OA$ объекта $o \in O$ называется валидируемым в алфавите A, если политика безопасности обеспечивает выполнение условия $AV(s, q) \in A^*$.

Параметры типа $t \in TA$ называются контролируемыми по имени, если политика безопасности обеспечивает выполнение условия $EAT(o, t) = EAT(s, t)$.

Аутентификатором объекта $o \in O$ будем называть значение de_jure_auth, вычисленное по атрибутам объекта o в соответствии с методом вычисления функции $assign\_auth$.

Аутентификатором субъект-сессии $s \in S$ будем называть значение de_facto_auth, вычисленное по атрибутам субъект-сессии s в соответствии с методом вычисления функции $assign\_auth$.

В рамках предложенной модели опишем метод построения аутентификатора, обеспечивающего выполнение следующих требований политики безопасности: базовое управление доступом пользователя к ресурсам, контроль целостности имён параметров и их значений, валидация значений параметров в заданном алфавите. Идея метода заключается в построении аутентификатора — строки, содержащей конкатенацию всех контролируемых атрибутов субъект-сессии, объекта и метода доступа, — и последующем его использовании в криптографическом протоколе аутентификации сообщений.

Метод. Условия применения метода: определены сущность $e \in E$ и функция $assign\_auth$, задающая политику построения аутентификатора сущности.

1) Построить список L атрибутов сущности e, соответствующих параметрам. Перейти к его началу.
2) Если параметры из списка L являются контролируемыми по имени, то положить $auth = auth||i_1|| \ldots ||i_m$, где $\{i_1, \ldots, i_m\}$ — упорядоченное множество имён

атрибутов и $EAT(e,t) = \{i_1, \ldots, i_m\}$ для типа t, соответствующего типу «параметр».

3) Для каждого контролируемого по значению параметра $l \in L$ положить $auth = auth||l||AV(e,l)$ и удалить его из L.
4) Для каждого валидируемого параметра $l \in L$ положить $auth = auth||l||([AV(e,l)] \setminus A)$ и удалить его из L.
5) Положить $auth = auth||id_s||id_r||op$, где id_r — идентификатор объекта; id_s — атрибут-идентификатор пользователя; op — метод доступа.
6) Выполнить протокол аутентификации сообщений, используя значение $auth$ в качестве одного из его параметров.

Данный метод аутентификации HTTP-сообщений может быть реализован на основе криптографических протоколов с ключевыми хеш-функциями.

Особенности функционирования веб-приложений предъявляют следующие требования к протоколу: протокол должен быть прозрачен для клиента (нежелательна его реализация на клиентской стороне); протокол должен быть двухшаговым; протокол может быть реализован в рамках взаимодействия веб-браузера и веб-сервера.

Параметры протокола: k — долговременный ключ сервера; k_r — одноразовый случайный ключ сервера; id_r — идентификатор защищаемого объекта; id_s — идентификатор пользователя; $time$ — текущее значение времени; L_P — запись политики P на некотором языке L. Действия протокола следующие:

1) Пользователь инициализирует отправку HTTP-запроса в рамках субъект-сессии $s \in S$ с атрибутом-идентификатором пользователя id_s.
2) Сервер по полученному запросу формирует ответ, содержащий объект $o \in O$; в соответствии с политикой безопасности P для объекта o вычисляет значение de_jure_auth, значения атрибутов $mac = h(k_r, de\_jure\_auth, time)$ и $policy = E_k(L_P, time, k_r)$; отправляет значения $policy$ и mac вместе с описанием объекта o в HTTP-ответе.
3) Пользователь в рамках субъект-сессии s отправляет запрос на доступ вида $op \in OP$ к объекту o вместе с атрибутами $policy$ и mac.
4) Сервер по атрибутам субъект-сессии s получает значения LP, $time$ и k_r, вычисляет значение de_facto_auth, находит $mac' = h(k_r, de\_facto\_auth, time)$, проверяет совпадение mac и mac', а также соответствие временной метки допустимому интервалу.

Особенность базового протокола заключается в том, что никакие данные, кроме главного ключа k, на сервере не хранятся. Это позволяет реализовать stateless-механизм и обойтись без поддержки устойчивых (persistence) соединений и без хранения политики безопасности в общедоступной сессии (sharing session) при реализации протокола для веб-фермы.

ЛИТЕРАТУРА

1. *Черемушкин А. В.* Криптографические протоколы. Основные свойства и уязвимости: учеб. пособие для студ. учреждений высш. проф. образования. М.: Издательский центр «Академия», 2009. 272 с.
2. Reducing web application attack surface. `http://blog.spiderlabs.com/2012/07/reducing-web-apps-attack-surface.html`
3. Signing and Authenticating REST Requests. `http://docs.aws.amazon.com/AmazonS3/latest/dev/RESTAuthentication.html`

4. Facebook developers reference. https://developers.facebook.com/docs/reference/php/facebook-getSignedRequest
5. *Barth A., Jackson C., and Mitchell J.* Robust defences for cross-site request forgery // Proc. 15th ACM Conf. on Computer and Communications Security. ACM Press, 2008. P. 75–87.
6. ModSecurity Advanced Topic of the Week: HMAC Token Protection. http://blog.spiderlabs.com/2014/01/modsecurity-advanced-topic-of-the-week-hmac-token-protection.html
7. Understanding ASP.NET View State. http://msdn.microsoft.com/library/ms972976.aspx
8. NIST 800-162. Guide to Attribute Based Access Control (ABAC) Definition and Considerations. http://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.sp.800-162.pdf

УДК 004.94

ОБ ИНФОРМАЦИОННЫХ ПОТОКАХ ПО ВРЕМЕНИ, ОСНОВАННЫХ НА ЗАГОЛОВКАХ КЭШИРОВАНИЯ ПРОТОКОЛА HTTP

Д. Н. Колегов, О. В. Брославский, Н. Е. Олексов

Рассматриваются информационные потоки по времени через заголовки кэширования протокола HTTP. Приводятся практические примеры реализации данных потоков и их основные характеристики, в частности достижимость на практике максимальной пропускной способности таких каналов при достаточно высоком уровне точности — 99,8 %.

Ключевые слова: *компьютерная безопасность, скрытые каналы, HTTP.*

Рассматривается задача анализа и реализации информационных потоков по времени, основанных на заголовках кэширования протокола HTTP. Обнаружение информационных потоков по времени и соответствующих им скрытых каналов в компьютерных системах является одной из задач компьютерной безопасности [1]. Распространённость протокола HTTP делает выявление методов реализации информационных потоков по времени перспективным направлением для исследований.

Известные на данный момент информационные потоки в протоколе HTTP, как правило, основываются на отправке HTTP-запросов со специальными GET- или POST-параметрами или на применении стеганографических методов для сокрытия факта передачи информации в HTTP-заголовках. Однако данные методы меняют стандартную структуру HTTP-запроса, а значит, требуют соответствующей модификации веб-сервера. Информационные потоки, рассматриваемые в данной работе, не накладывают дополнительных ограничений на конфигурацию веб-сервера и, следовательно, представляют больший интерес для изучения.

Заголовки кэширования в протоколе HTTP хранят информацию о времени последнего изменения веб-страницы, тем самым позволяя клиенту не загружать веб-страницу, если она не была изменена с момента последнего запроса. Таким образом, возможна передача информации на основании данных об изменениях запрашиваемой веб-страницы. Например, ситуация, при которой некоторая веб-страница была изменена с момента последнего обращения к ней, может быть интерпретирована как получение одного бита информации.

Рассмотрим общую схему информационного потока по времени данного типа (рис. 1). Пусть O_1 — объект, доступный на чтение процессу S_1 на $host_1$; O_2 — веб-ресурс, расположенный на $host_1$; O_3 — HTTP-ответ; O_4 — объект, доступный на запись