D.N. Kolegov, O.V. Broslavsky, N.E. Oleksov

Researching of Covert Timing Channels Based on HTTP Cache Headers in Web API

Keywords: covert channels, HTTP, Web API, web application security.
In this paper, it is shown how covert timing channels based on HTTP cache headers can be implemented using different Web APIof Google Drive, Dropbox and Facebook Internet services.

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

ИССЛЕДОВАНИЕ СКРЫТЫХ КАНАЛОВ ПО ВРЕМЕНИ НА ОСНОВЕ ЗАГОЛОВКОВ КЭШИРОВАНИЯ ПРОТОКОЛА HTTPB WEBAPI

Скрытым каналом называется непредусмотренный разработчиком системы информационных технологий и автоматизированных систем коммуникационный канал, который может быть применен для нарушения политики безопасности [1, 2]. Скрытые каналы, как правило, рассматриваются в контексте следующего класса угроз:

- скрытая передача вредоносных программ и данных на контролируемые нарушителем компоненты информационной системы;
- реализация коммуникационных каналов управления бот-сетями и агентами (zombie);
- скрытая реализация протоколов распределения криптографических ключей.

По механизму передачи информации скрытые каналы разделяют на скрытые каналы по памяти и скрытые каналы по времени. Скрытые каналы по памяти основаны на наличии памяти, в которую передающий субъект записывает информацию, а принимающий – считывает ее. Скрытые каналы по времени так или иначе используют фактор времени и, как правило, предполагают, что передающий информацию субъект модулирует с помощью передаваемой информации некоторый изменяющийся во времени процесс, а субъект, принимающий информацию, в состоянии демодулировать передаваемый сигнал, наблюдая несущий информацию процесс во времени [2]. По направлению передачи скрытые каналы дополнительно классифицируются на однонаправленные (unidirectional) и двунаправленные (bidirectional).

В настоящее время одной из актуальных задач в области безопасности веб-приложений является идентификация и анализ скрытых каналов по времени. Данная задача возникает в связи с возможностью использования нарушителем скрытых каналов для передачи команд и данных от серверов управления нарушителя к захваченным и контролируемым веб-браузерам пользователей [3]. Большинство известных скрытых каналов созданы на протоколе HTTP, чаще всего являются скрытыми каналами по памяти и основаны на отправке HTTP-сообщений определенного типа с заданными параметрами, несущими информацию в теле запроса или ответа, а также на применении стеганографических методов [4]. Исследованию скрытых каналов по времени, допускающих эксплуатацию в современных веб-приложениях, уделяется недостаточное внимание. Поэтому в настоящее время авторами активно исследуется класс скрытых каналов по времени, основанных на заголовках кэширования протокола HTTP[5, 6]. Данные скрытые каналы являются однонаправленными и позволяют передавать данные от сервера к клиенту в сети Интернет со скоростью до 15 бит/с. В данной работе исследуется возможность реализации данных скрытых каналов на основе интерфейсов программирования веб-приложений (WebAPI) популярных веб-сервисов Интернет.

Скрытые каналы по времени на основе заголовков кэширования HTTP. Данные скрытые каналы были предложены и детально описаны в [5]. Рассмотрим общую схему функционирования данных скрытых каналов в терминах субъектно-сущностных моделей управления доступом и диаграмм потоков данных.

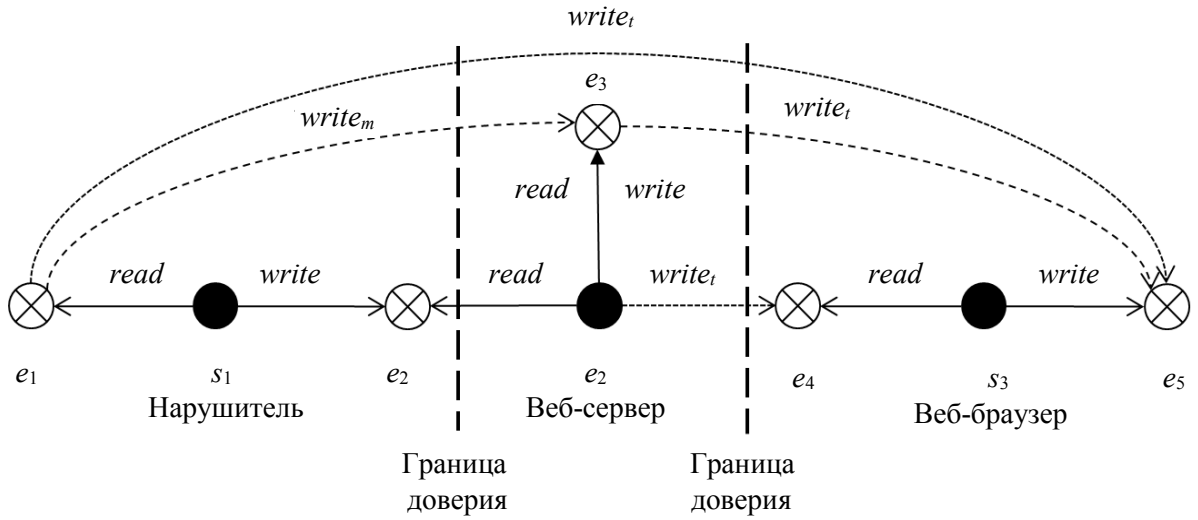

Рис. 1. Общая схема скрытого канала по времени

Пусть e_1 – сущность-файл, расположенная на стороне нарушителя и доступная на чтение субъекту s_1; e_3 – сущность-ресурс, расположенная на стороне веб-сервера и доступная на запись субъекту s_1 через некоторые интерфейсы веб-сервера s_2; e_2 – сущность HTTP-запрос; e_4 – сущность HTTP-ответ; e_5 – сущность-файл, доступная на запись процессу s_3 на стороне клиента.

Пусть субъекту s_1 требуется передать данные субъекту s_3. Тогда в каждый момент времени s_1, в зависимости от значения бита, считанного из сущности e_1, осуществляет доступ на запись к ресурсу e_3. Субъект s_1 не обязательно должен располагаться на веб-сервере, важно, чтобы он имел возможность осуществлять изменение данных в e_3 на основе данных из e_1. Субъект-процесс s_3 выполняет HTTP-запрос e_2 к ресурсу e_3 и после получения HTTP-ответа e_4, отображающего изменения сущности e_3, записывает в сущность e_5 бит, соответствующий выбранному коду. Данная схема информационного потока позволяет выделить два базовых сценария реализации скрытых каналов по времени.

Сценарий M_1. Субъектами-нарушителями являются субъекты s_1 и s_3, субъект s_2 – доверенный веб-сервер и не контролируется нарушителем. Предполагается, что нарушитель не может напрямую изменять содержимое HTTP-ответа e_4, но, имея доступ на запись к ресурсу e_3, нарушитель может путем реализации информационных потоков по памяти от e_1 к e_3 менять время последней модификации ресурса e_3 и, тем самым, косвенно менять содержимое HTTP-ответа e_4. При этом субъект s_1 не имеет никакой информации о HTTP-запросах s_3 к s_2, т.е. s_1 и s_3 функционируют независимо друг от друга, руководствуясь только временными интервалами. Данный интервал одинаков для субъекта-отправителя и субъекта-получателя и является параметром, выбираемым обеими сторонами на основании пропускной способности сети. Стоит отметить, что такой скрытый канал обеспечивает анонимность для субъектов-нарушителей: средства кон-

троля, функционирующие между границами доверия, не могут обнаружить сетевое взаимодействие субъектов-нарушителей s_1 и s_3.

Сценарий M_2. Субъектами-нарушителями являются субъекты s_1, s_2 и s_3. При этом субъект s_2 является веб-сервером, контролируемым нарушителем, и может произвольно модифицировать сущность-ответ e_4 на основе данных из e_3. Такие скрытые каналы не обеспечивают свойства анонимности, но более просты в реализации и обладают большей пропускной способностью.

Описанные выше сценарии различаются лишь возможностями нарушителя по контролю веб-сервера и не накладывают никаких ограничений на способы модулирования полезной информации через заголовки кэширования HTTP. Рассматриваемые скрытые каналы могут быть реализованы на основе сущностей HTTP-date или entity-tag.

Стоит отметить, что кэширующие прокси-серверы или механизмы кэширования веб-браузеров, как правило, не могут быть использованы для блокирования данных скрытых каналов по времени. Действительно, в рамках сценария M_2 нарушитель полностью контролирует веб-сервер, поэтому он может обеспечить уникальность URL всех запрашиваемых ресурсов (например, с помощью URLrewriting) или дополнительно установить заголовки, запрещающие выполнять кэширование ответов (например, Cache-Control: no-store). В рамках сценария M_1 нарушитель не контролирует заголовки кэширования, устанавливаемые веб-сервером, поэтому в случае использования кэширующих прокси-серверов здесь необходимо выбирать веб-сервисы, запрещающие кэширование (например, GoogleDrive).

Реализация скрытых каналов по времени с использованием облачных файловых сервисов. Основным недостатком сценария M_1 является крайне низкая пропускная способность реализуемого скрытого канала. Наиболее распространенным временем обновления заголовков Last-Modifiedи ETag является 1 с, причем в сценарии M_1 нарушитель не может изменить частоту обновления заголовка. Поэтому пропускная способность скрытых каналов, использующих стандартные веб-серверы (например, Apacheили Nginx), не превосходит 1 бит/с. В то же время использование в качестве s_2 доверенного веб-сервера с большей частотой обновления заголовков кэширования реализуемой логикой веб-приложения позволяет увеличить пропускную способность скрытых каналов.

Примерами таких веб-серверов могут быть облачные файловые хостинги GoogleDriveиDropbox. Повышенная частота обновления заголовков кэширования вызвана необходимостью поддержания актуальной информации о размещенных на хостинге файлах. Большинство крупных облачных файловых сервисов предоставляют программный интерфейс (API) для управления уже размещенной на сервере информацией: загрузки и выгрузки файлов, их систематизации и обновления метаинформации (в том числе и времени последнего изменения файла). Таким образом, при помощи предоставляемого сервисом API возможна реализация скрытого канала на основе заголовков кэширования с использованием сценария M_1.

Рассмотрим реализацию скрытого канала на платформе облачного файлового хостинга GoogleDrive. Для этого был размещен файл e_3, доступный на запись субъекту s_1 и на чтение субъекту s_3, сами субъекты располагались на различных узлах сети Интернет. Для передачи одного бита информации субъект s_1 при помощи POST-запроса к API по адресу *http://www.googleapis.com/drive/v2/files/file/touch* изменяет время последнего доступа к сущности e_3 с идентификатором *file*. Для получения переданного бита субъект s_3 также совершает запрос к API по адресу *http://www.googleapis.com/drive/v2/files/file* и получает значение используемого заголов-

ка ETag из метаинформации о файле, после чего восстанавливает переданный бит, руководствуясь описанными выше алгоритмами. При тестировании данный скрытый канал показал пропускную способность 3 бит/с при точности передачи (количестве бит, переданных без ошибок, к общему числу переданных бит) 99,8%.

Другой крупный облачный хостинг Dropbox, в отличие от Google Drive, не предоставляет возможности обновлять время последнего изменения файла непосредственно при помощи запроса к API. Кроме того, метаинформация о файле, хранящемся на серверах Dropbox, содержит только значение времени последней модификации файла в виде «Wed, 20 Jul 2011 22:04:50 +0000» и не содержит значения ETag. Однако данного значения достаточно для реализации скрытого канала, но с максимальной пропускной способностью, не превышающей 1 бит/с.

Так как обновлять время последнего изменения файла непосредственно на серверах Dropbox не представляется возможным, то единственным вариантом остается повторная загрузка обновленного файла на облачный хостинг. Идентификатором файла в системе Dropbox является полный путь файла в системе, таким образом, повторная загрузка файла с тем же именем по тому же пути приведёт к обновлению хранящегося в системе файла. Ввиду ограничения максимальной пропускной способности в 1 бит/с такой вариант вполне допустим и не накладывает дополнительных ограничений на пропускную способность при использовании файла достаточно маленького размера. Достаточность здесь определяется пропускной способностью интернет-канала от субъекта s_1 до сервера Dropboxs_2.

Функционирование рассматриваемого скрытого канала в рамках облачного хостинга Dropbox возможно следующим образом. Субъекты s_1 и s_3 авторизованы в системе и имеют необходимыетокены доступа OAuth2. Субъект s_1 через заданные промежутки времени осуществляет чтение одного бита из сущности e_1 и в зависимости от полученных данных совершает или не совершает PUT-запрос e_2 по адресу *https://api-content.dropbox.com/1/files_put/auto/file?overwrite=true*, тем самым обновляя файл с идентификатором *file*. В тот же временной интервал субъект s_3 выполняет GET-запрос по адресу *https://api.dropbox.com/1/metadata/auto/file* и получает ответ e_4, содержащий всю метаинформацию файла *file*, включая время его последнего изменения. Субъект s_3 записывает в e_5 бит 1, если *file* был модифицирован с момента последнего запроса, и бит 0 в противном случае.

Описанный вариант функционирования скрытого канала на платформе Dropbox не является единственным. Получаемый ответ e_4 содержит также поле *hash*, значение которого вычисляется как хеш-функция от всех прочих полей получаемой в ответе метаинформации. После чего значение поля *hash* может быть указано в качестве параметра запроса к *https://api.dropbox.com/1/metadata/auto/file*, и, если значение полей метаинформации о файле не изменилось, ответ e_4 будет содержать только код 304(Not Modified). Таким образом, функционирование скрытого канала аналогично скрытому каналу при помощи заголовка If-Modified, описанному в [5].

Максимальная пропускная способность описанных выше скрытых каналов по времени, использующих Dropbox, равна 1 бит за (L+S+U) с, где L – время, необходимое s_3 для выполнения запроса к e_3, S – время, уходящее на обработку запроса на серверах хостинга, U – время, необходимое s_1 для обновления файла на облачном хостинге. Данное время, конечно, не может превышать максимальной теоретической пропускной способности в 1 бит/с.

Реализация скрытых каналов по времени с использованием FacebookGraphAPI. Механизмы кеширования протокола HTTP используют не только облачные файловые сервисы и веб-сайты, но и множество веб-приложений. Например,

социальная сеть Facebook использует заголовок кеширования ETag при работе с Graph API. Каждый ответ на запрос к Graph API содержит значение заголовка ETag, которое может быть использовано при последующих запросах. Если указать данное значение ETag в заголовке запроса If-None-Match, то ответ будет получен, только если запрашиваемая информация была изменена, в противном случае ответ будет содержать только сообщение 304(Not Modified).

Таким образом, для реализации скрытого канала на основе заголовка ETag достаточно любого запроса API, на результат которого сможет влиять субъект s_1. В каждый момент времени s_1 при необходимости изменяет результат выполнения выбранного запроса к API, а субъект s_3 при помощи данного запроса получает соответствующее значение ETag, после чего руководствуется логикой модулирования данных скрытого канала [5].

Для примера рассмотрим запрос к API, получающий информацию обо всех записях новостной ленты конкретного пользователя. Для реализации данного скрытого канала необходимо приложение Facebook, имеющее доступ на чтение и изменение новостной ленты, а также профиль пользователя, разрешившего доступ данному приложению. Приложение необходимо для получения токена доступа, позволяющего оперировать с запросами к API. Профиль пользователя же в данном случае выступает в качестве сущности e_3.

В каждый временной интервал для передачи одного бита информации субъект s_1 совершает (или не совершает, в зависимости от значения передаваемого бита) POST запрос e_2по адресу*http://graph.facebook.com/v2.3/page/feed*. Данный запрос обновляет содержимое ленты пользователя с идентификатором *page*, создавая новую запись от имени пользователя. В этот же временной интервал субъект s_3 совершает GET запрос на адрес *http://graph.facebook.com/v2.3/page/feed*, получает ответ e_4, содержащий все записи пользователя, а также значение ETag. Основываясь на полученном значении заголовка ETag, субъект s_3 принимает решение о полученном бите информации.

Так как GraphAPI поддерживает также и ответ на запрос, содержащий заголовок If-None-Match, то возможна также и реализация скрытого канала на основе этого заголовка. Для этого s_3 при GET запросе к *http://graph.facebook.com/v2.3/page/feed* указывает также заголовок If-None-Match, содержащий последнее полученное значение заголовка ETag, а вывод о полученном бите делает, основываясь на коде состояния HTTP ответа – 200(OK) или 304(Not Modified).

В данном методе для изменения состояния ленты пользователя использовалось добавление новых записей от имени пользователя, однако это не обязательное условие. Предоставляемый интерфейс API дает возможность также обновлять или удалять старые записи, что позволяет имитировать поведение типичного пользователя социальной сети, тем самым обеспечивая дополнительную скрытность. Кроме того, как уже было упомянуто ранее, в роли сущности e_3 может выступать не только новостная лента пользователя, но и лента публичной страницы, альбомы пользователей и публичных страниц, личные сообщения пользователя и другие сущности, изменение которых возможно при помощи Graph API.

Практическая пропускная способность скрытых каналов при помощи FacebookGraph API описывается выражением 1 бит за (L+S) с, что при проведении экспериментов позволило реализовать анонимные скрытые каналы с пропускной способностью 1,25 бит/с.

СПИСОК ЛИТЕРАТУРЫ:

1. CWE-514. Covert Channel. URL: https://cwe.mitre.org/data/definitions/514.html.
2. ГОСТ Р 53113.1 – 2008. Информационная технология. Защиты информационных технологий и автоматизированных систем от угроз информационной безопасности, реализуемых с использованием скрытых каналов. Ч.1. Общие положения.
3. Alkorn W., Frichot C., and Orru M. The Browser Hacker's Handbook. Indianapolis: John &Wiley Sons, 2014. 648p.
4. BrownE., YuanB., JohnsonD., andLutzP. CovertchannelsintheHTTPnetworkprotocol: Channelcharacterization and detecting Man-in-the-Middle attacks // Proc. 5th Intern. Conf. Inform. Warfare and Security. Ohio, USA, April 8-9. The Air Force Institute of Technology, 2010. P. 56-65.
5. Колегов Д.Н., Брославский О.В., Олексов Н.Е. Скрытые каналы по времени на основе заголовков кэширования протокола HTTP // Прикладаня дискретная математика. 2015. №28. С. 71-85.
6. The Browser Exploitation Framework Project [Электронныйресурс]. URL: http://beefproject.com.

REFERENCES:

1. CWE-514. Covert Channel. URL: https://cwe.mitre.org/data/definitions/514.html.
2. GOST R 53113.1 – 2008. Information technology. Protection of Information Technologies and Automated Systems against Information Secutiy Threats Implemented using Covert Channels.P.1. Common Statements.
3. Alkorn W., Frichot C., and Orru M. The Browser Hacker's Handbook. Indianapolis: John &Wiley Sons, 2014. 648p.
4. BrownE., YuanB., JohnsonD., andLutzP. CovertchannelsintheHTTPnetworkprotocol: Channelcharacterization and detecting Man-in-the-Middle attacks // Proc. 5th Intern. Conf. Inform. Warfare and Security. Ohio, USA, April 8-9. The Air Force Institute of Technology, 2010. P. 56-65.
5. Kolegov D.N., Broslavssky O.V., Oleksov N.E. CovertTimingChannelsbasedonHTTPCacheHeaders // Applied Discrete Mathematics. 2015.№28. P. 71-85.
6. TheBrowserExploitationFrameworkProject. URL: http://beefproject.com.