Д. Н. Колегов, О. В. Брославский, Н. Е. Олексов, Скрытые каналы по времени на основе заголовков кэширования протокола HTTP, *ПДМ*, 2015, номер 2, 71–85

DOI: http://dx.doi.org/10.17223/20710410/28/8

УДК 004.94

СКРЫТЫЕ КАНАЛЫ ПО ВРЕМЕНИ НА ОСНОВЕ ЗАГОЛОВКОВ КЭШИРОВАНИЯ ПРОТОКОЛА HTTP

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

Национальный исследовательский Томский государственный университет, г. Томск, Россия

Описано неизвестное ранее семейство скрытых каналов по времени протокола HTTP на основе заголовков кэширования, построены их общая схема функционирования и две граничные модели угроз, в рамках которых предложены реализации и экспериментальные оценки рассматриваемых скрытых каналов. Показано, каким образом скрытые каналы данного семейства могут быть реализованы в веб-браузерах. Проведено сравнение обнаруженных скрытых каналов по времени на основе заголовков кэширования протокола HTTP с другими известными механизмами реализации скрытых каналов по времени в веб-браузерах. Разработаны программные компоненты для инструментального средства анализа защищённости BeEF, определяющие возможность реализации рассматриваемых скрытых каналов по времени в информационной системе.

Ключевые слова: *компьютерная безопасность, анализ защищённости, HTTP, информационные потоки, скрытые каналы, безопасность веб-приложений, безопасность веб-браузеров, бот-сети.*

DOI 10.17223/20710410/28/8

COVERT TIMING CHANNELS OVER HTTP CACHE-CONTROL HEADERS

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

National Research Tomsk State University, Tomsk, Russia

E-mail: d.n.kolegov@gmail.com, o.v.broslavsky@gmail.com, n.e.oleksov@gmail.com

We introduce and discuss a new family of timing covert channels based on HTTP cache headers. We propose a general scheme of the timing covert channels in terms of access control models and data flow diagrams and suggest two base threat models for them. We then consider peculiarities of program implementation of the timing covert channels and their bandwidth depending on a HTTP cache header, a threat model, a programming language (C, JavaScript, Python, Ruby), and an environment. Finally we provide the basic characteristics of the implemented covert channels in web browsers and BeEF.

Keywords: *computer security, HTTP, cache-control headers, covert channels, web application security, web browsers security, botnets.*

Введение

Впервые скрытые каналы были описаны как механизмы сокрытия вредоносных данных внутри разрешённых данных, передаваемых по некоторому коммуникационному каналу [1]. В соответствии с нормативными документами *скрытым каналом* (covert

channel) называется не предусмотренный разработчиком системы информационных технологий и автоматизированных систем коммуникационный канал, который может быть применён для нарушения политики безопасности [2, 3].

Если первоначально в компьютерной безопасности скрытые каналы рассматривались в основном как источник угрозы нарушения политики безопасности мандатного управления доступом типа LBAC [4], приводящей к реализации запрещённых информационных потоков от сущностей с высоким уровнем конфиденциальности к сущностям с низким уровнем конфиденциальности, то в настоящее время скрытые каналы рассматриваются в контексте более широкого класса угроз [2, 5, 6], включающего в себя угрозы:

— скрытой передачи вредоносных программ и данных на контролируемые нарушителем компоненты информационной системы;
— передачи нарушителем команд управления агентам для выполнения;
— организации коммуникационных каналов управления бот-сетями;
— скрытой передачи криптографических ключей и параметров функционирования.

Традиционно по механизму передачи информации скрытые каналы разделяют на скрытые каналы по памяти и скрытые каналы по времени. Как правило, *скрытые каналы по памяти* основаны на наличии памяти, в которую передающий субъект записывает данные и из которой принимающий субъект считывает эти же данные. *Скрытые каналы по времени*, как правило, предполагают, что передающий субъект на основе данных модулирует некоторый изменяющийся во времени процесс, а принимающий субъект в состоянии демодулировать передаваемые данные, наблюдая изменение этого процесса во времени. Считается, что обнаружение скрытых каналов по времени является более сложной задачей, чем обнаружение скрытых каналов по памяти.

Для иллюстрации скрытого канала по времени рассмотрим следующий классический пример. Пусть в операционной системе имеются два процесса s_1 и s_2, которые могут создавать «слушающий» сокет на некотором заранее выбранном порту. Если процессу s_1 необходимо передать бит, равный 1, процессу s_2, то процесс s_1 создает «слушающий» сокет, если же необходимо передать бит, равный 0, то процесс s_1 ничего не делает. Для получения данных от процесса s_1 процесс s_2 пытается создать слушающий сокет на том же порту. Если при создании сокета субъектом s_2 в системе возникла ошибка, то процессом s_1 был передан бит 1, иначе — бит 0. Таким образом, процессы s_1 и s_2 с помощью рассмотренного скрытого канала, использующего механизм сокетов, реализуют в системе информационный поток, который является разрешённым или запрещённым в зависимости от политики безопасности информационной системы.

В настоящее время веб-приложения являются неотъемлемой частью систем информационных технологий и автоматизированных систем управления. В связи с этим проблема анализа защищённости веб-приложений является одной из самых актуальных в практической компьютерной безопасности. Одной из особенностей веб-приложений является использование в качестве клиента веб-браузера. Современные веб-браузеры являются наиболее распространённым, универсальным и широкоиспользуемым, но в то же время уязвимым программным обеспечением [7], что позволяет их использовать в различного рода атаках на информационые системы. Основным механизмом безопасности веб-браузеров является механизм Same Origin Policy. Обход данного механизма позволяет злоумышленнику выполнить произвольный код Javascript в веб-браузере пользователя в контексте (origin) уязвимого веб-приложения. Таким образом, после захвата контроля над веб-браузером возникает задача передачи управляю-

щей (сигнальной) информации от серверов управления нарушителя к веб-браузерам пользователей. Одним из способов передачи такой информации могут быть и скрытые каналы. Данные коммуникационные каналы, как правило, используются в бот-сетях и во вредоносном программном обеспечении.

Основным протоколом веб-приложений является протокол HTTP. Перечислим основные механизмы и элементы, используемые для реализации скрытых каналов в протоколе HTTP [7, 8]:

— перенаправление (redirect);
— HTTP cookies;
— заголовок HTTP Referer;
— произвольные (custom) HTTP-заголовки;
— коды ответов;
— пути и параметры, передаваемые в URL.

Большинство известных скрытых каналов в протоколе HTTP являются скрытыми каналами по памяти и основаны на отправке HTTP-сообщений определённого типа с заданными параметрами, несущими полезную информацию в теле запроса или ответа (HTTP-туннели), или на применении стеганографических методов для сокрытия факта передачи информации в HTTP-сообщениях.

Например, скрытый канал по памяти от клиента к серверу может быть реализован через заголовок кэширования *If-Range* следующим образом: передаваемая клиентом информация представляется в шестнадцатеричном виде и отправляется в значении заголовка *If-Range*. Так, передаваемое сообщение «hello» будет передано в виде «If-Range: 120c7bL-32bL-68656c6c6fL», где «68656c6c6f» — передаваемое сообщение, а «120c7bL-32bL» — стандартные части значения *ETag*-заголовка.

Примером скрытого канала с использованием стеганографических методов в HTTP-заголовках является скрытый канал через заголовок *Accept-Language*, изначально предназначенный для сообщения веб-серверу о поддерживаемых клиентом естественных языках. Предположим, что значение заголовка *Accept-Language*, равное «*en*», будет интерпретироваться сервером как 0, а «*fr*» — как 1. Тогда клиент может передать на сервер значение 0x50 путём отправки заголовка *Accept-Language* со значением «$en, fr, en, fr, en, en, en, en$».

Вместе с тем использование стеганографических методов, как правило, меняет стандартные значения заголовков или структуру HTTP-сообщений и, как следствие, приводит к обнаружению скрытого канала средствами фильтрации и контроля. Кроме того, для таких скрытых каналов может потребоваться модификация веб-сервера, позволяющая корректно обрабатывать недопустимые с точки зрения протокола значения заголовков.

В данной работе предлагаются скрытые каналы, использующие HTTP-заголовки по их прямому назначению, не вносящие дополнительных изменений в структуру заголовков и в формат передаваемых данных, а потому неотличимые от потока разрешённых HTTP-сообщений веб-приложения.

Обнаружению скрытых каналов по времени в протоколе HTTP уделяется мало внимания. Более того, авторам не известны описанные в научной литературе скрытые каналы по времени для протокола HTTP. Скрытые каналы, предложенные в работах [7 – 9], по своей сути являются скрытыми каналами, использующими механизмы функционирования протоколов IP, TCP или DNS, доступные через API веб-браузера, но не механизмы и структуры данных самого протокола HTTP.

В сетевых протоколах принято взаимодействующих субъектов разделять на субъектов-клиентов (или просто клиентов), которые инициируют соединение и отправляют запросы, и субъектов-серверов (или просто серверов), которые ожидают соединения, производят обработку запроса и возвращают клиентам соответствующие ответы. Таким образом, скрытые каналы в сетевых протоколах могут быть дополнительно классифицированы по направлению передачи информации на каналы от сервера к клиенту и от клиента к серверу, а также по возможности передачи данных — на однонаправленные (unidirectional) и двунаправленные (bidirectional).

Поскольку в протоколе HTTP именно клиент является инициатором соединения, реализация скрытых каналов, передающих информацию от клиента к серверу, является более простой и эффективной. В данной работе, напротив, пойдёт речь о скрытых каналах по времени, реализуемых от сервера к клиенту в протоколе HTTP, ввиду их большей ценности для исследований.

1. Заголовки кэширования протокола HTTP

Заголовком в протоколе HTTP называется пара вида «имя : значение». В HTTP-сообщениях заголовки отделяются друг от друга пустой строкой и предназначены для передачи служебной информации между клиентом и сервером. Например, заголовок *User-Agent* сообщает веб-серверу информацию о типе и версии веб-клиента, а заголовок *Accept-Language* содержит информацию о поддерживаемых веб-клиентом языках. Заголовки протокола HTTP делятся на заголовки запросов и заголовки ответов.

Заголовки запроса — это заголовки, используемые веб-клиентом для сообщения веб-серверу дополнительной информации (например, заголовок *Accept-Encoding*) или для уточнения характера запрашиваемой информации (например, заголовок *Content-Range*). *Заголовки ответа* — это заголовки, используемые веб-сервером для описания ответа (например, заголовок *Content-Length*, содержащий длину тела ответа, или заголовок *Last-Modified*, содержащий дату последнего изменения документа).

Кэширование является одним из основных механизмов протокола HTTP [10]. Его цель — устранение необходимости повторной отправки запросов и ответов, содержащих уже имеющиеся данные. Кэширование позволяет сократить время повторной загрузки веб-страницы, что приводит к увеличению производительности и доступности веб-приложений. Заголовки кэширования протокола HTTP позволяют клиенту и серверу обменяться информацией о наличии и актуальности запрашиваемых ресурсов.

Рассмотрим основные заголовки кэширования протокола HTTP, используемые в данной работе. Заголовки ответа, сообщающие клиенту о состоянии запрошенного ресурса:

1) *Last-Modified* — содержит дату последнего изменения запрошенного клиентом ресурса в формате «Tue, 10 Apr 2014, 12:34:56 GMT».
2) *ETag* (*entity-tag*) — содержит идентификатор запрошенного клиентом ресурса. Среди наиболее распространенных веб-серверов только в Apache стандартизован алгоритм формирования заголовка *ETag* [11]. Значение *ETag* в соответствии с данным алгоритмом формируется из шестнадцатеричных значений идентификатора ресурса (*inode*), его размера и времени последнего изменения в формате *mtime*: «ETag: 120c7bL-32bL-4f86d4105ac62L».

Заголовки запроса, сообщающие веб-серверу условия, при выполнении которых ему необходимо отправить клиенту изменённые ресурсы:

1) *If-Modified-Since*: HTTP-date;
2) *If-Unmodified-Since*: HTTP-date;

3) *If-Match*: entity-tag;
4) *If-None-Match*: entity-tag;
5) *If-Range*: HTTP-date или entity-tag.

Заголовки кэширования *If-Modified-Since* и *If-None-Match*, отправленные в запросе, позволяют определить, изменился ли ресурс на сервере. Сервер в ответ на такой запрос сообщает при помощи кода состояния HTTP-сообщения об изменении (200 OK) или о неизменении соответствующего ресурса (304 Not Modified). При этом в первом случае веб-сервер отправляет новую версию ресурса, а во втором ответ веб-сервера содержит только HTTP-заголовки.

Вторая пара заголовков *If-Unmodified-Since* и *If-Match* действует наоборот, спрашивая сервер, не изменился ли ресурс, и получая в ответ сообщение об изменении (412 Precondition Failed) или неизменении (200 OK).

Заголовок *If-Range* обычно используется в условных GET-запросах вместе с заголовками *If-Unmodified-Since*, *If-Match* или *Range* и позволяет уменьшить количество запросов, необходимых для получения новой полной версии запрашиваемого ресурса.

Таким образом, путём изменения ресурсов, запрашиваемых веб-клиентом, и получения отправляемых веб-сервером ответов возможна скрытая передача данных. Для этого достаточно принять следующую трактовку: изменение некоторого ресурса с момента последнего обращения к нему означает передачу бита 1, а его неизменение — бита 0. Рассмотрим возможные варианты реализации таких скрытых каналов, основанных на заголовках кэширования протокола HTTP.

2. Механизм функционирования скрытых каналов по времени на основе заголовков кэширования HTTP

В теории компьютерной безопасности *информационным потоком* от сущности-источника к сущности-приемнику называется преобразование данных в сущности-приемнике, осуществляемое субъектами информационной системы в зависимости от данных в сущности-источнике [12]. Рассмотрим представление информационного потока, порождаемого скрытым каналом на основе заголовков кэширования протокола HTTP, в рамках субъектно-сущностных схем управления доступом [12] и диаграмм потоков данных [13] (рис. 1).

Введём следующие обозначения на основе [12]: 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 — HTTP-запрос; e_4 — HTTP-ответ; e_5 — сущность-файл, доступная на запись процессу s_3 на стороне веб-клиента.

Пусть субъект s_1 хочет передать данные субъекту s_3. Тогда в каждый момент времени s_1 считывает один бит данных из сущности e_1 и, в зависимости от значения считанного бита, осуществляет или не осуществляет доступ на запись к ресурсу e_3. Важной особенностью является тот факт, что субъект s_1 не обязательно должен располагаться на веб-сервере, важно, чтобы он имел возможность осуществлять изменение данных в сущности e_3 на основе данных из сущности e_1 посредством сущностей-запросов e_2. Субъект-процесс s_3 выполняет HTTP-запрос к ресурсу e_3 и после получения HTTP-ответа e_4, отображающего изменения сущности e_3, пишет в сущность e_5 соответствующий бит (например, 1, если веб-страница была изменена, и 0 в противном случае).

Рассмотренная схема реализации информационного потока позволяет выделить две граничные модели угроз.

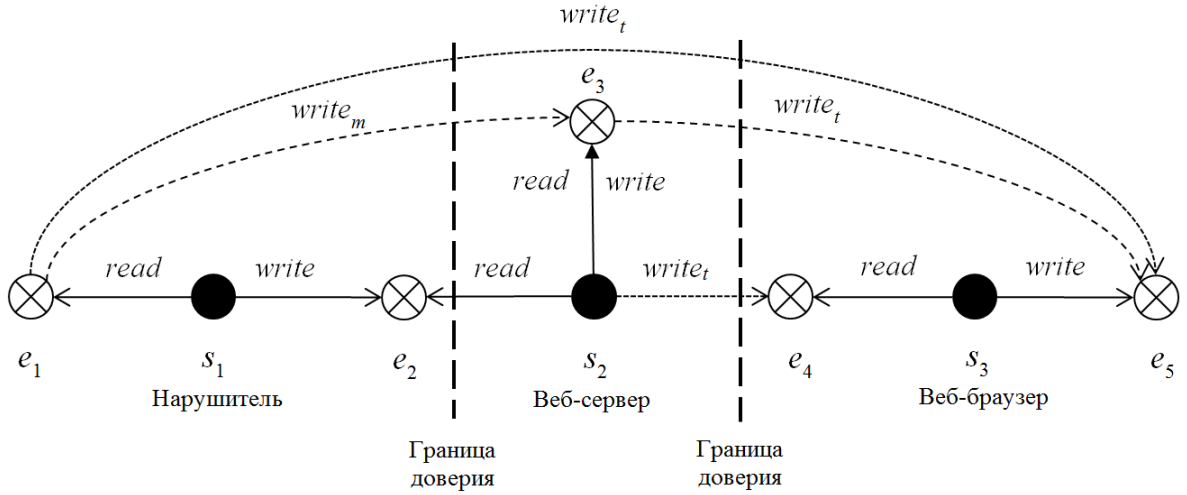

Рис. 1. Общая схема функционирования скрытых каналов по времени на основе заголовков кэширования HTTP

1) М о д е л ь M_1. Субъектами-нарушителями являются субъекты s_1 и s_3, субъект s_2 — доверенный веб-сервер (не контролируется нарушителем). Субъект-нарушитель s_1 является программным сценарием, имеющим возможность генерировать произвольные HTTP-запросы e_2 к интерфейсу веб-сервера s_2, зависящие от данных в e_1, а субъект-нарушитель s_3 является вредоносным сценарием JavaScript, функционирующим в веб-браузере на стороне пользователя. В рамках данной модели нарушитель не может напрямую изменять содержимое HTTP-ответа e_4, но, имея доступ на запись к ресурсу e_3, нарушитель может путём 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, то есть s_1 и s_3 функционируют независимо друг от друга, руководствуясь только временными интервалами. Данный интервал одинаков для процесса-отправителя и процесса-получателя и является параметром, выбираемым обеими сторонами на основании пропускной способности сети. В каждый такой временной интервал субъект s_1 осуществляет изменение сущности e_3, а субъект s_3, независимо от субъекта s_1, отправляет HTTP-запрос к e_3. В связи с этим возникает проблема синхронизации субъекта-отправителя s_1 и субъекта-получателя s_3. При этом неправильный выбор интервала приводит к увеличению ошибок при передаче в случае, когда время HTTP-запроса от s_3 к e_3 превышает выбранный интервал. Стоит отметить, что скрытый канал в модели M_1 обеспечивает анонимность для субъектов-нарушителей, т. е. средства контроля, функционирующие между границами доверия, не могут обнаружить сетевое взаимодействие субъектов-нарушителей s_1 и s_3.

2) М о д е л ь M_2. Субъектами-нарушителями являются субъекты s_1, s_2 и s_3. При этом субъект s_2 является веб-сервером, контролируется нарушителем и может произвольно модифицировать сущность-ответ e_4 на основе данных из e_3; s_3 — вредоносный сценарий JavaScript, функционирующий в веб-браузере на стороне пользователя. В данной модели взаимодействие субъектов-нарушителей s_1 и s_2 не имеет существенного значения. Для простоты можно считать, что в ней сущности e_1 и e_2 содержат одни и те же данные. Данный скрытый канал не обеспечивает свойства анонимности, но бо-

лее прост в реализации и обладает большей пропускной способностью. В этой модели также возникает задача обеспечения синхронизации, но уже между субъектами s_2 и s_3.

Рассматриваемые скрытые каналы по времени на основе заголовков кэширования протокола HTTP могут быть разделены на скрытые каналы на основе сущностей HTTP-date и на основе сущностей entity-tag. Отметим, что описанные выше модели угроз различаются лишь возможностями нарушителя по контролю веб-сервера и не накладывают никаких ограничений на способы модулирования полезной информации (payload) через заголовки кэширования протокола HTTP. Рассмотрим механизмы реализации, особенности и характеристики скрытых каналов каждой из групп в зависимости от используемой модели угроз.

3. Скрытые каналы по времени на основе сущностей HTTP-date

В механизмах кэширования протокола HTTP используются сущности HTTP-date в качестве меток времени. Например, заголовок *Last-Modified* содержит время последнего изменения ресурса, запрашиваемого клиентом: «Last-Modified: Tue, 10 Apr 2014, 12:34:56 GMT». Можно выделить следующие три варианта реализации скрытого канала по времени на основе сущностей HTTP-date.

1) На основе заголовка *Last-Modified*. Субъект s_3 обращается к сущности e_3 и получает HTTP-ответ e_4, содержащий начальное значение HTTP-date$_0$ заголовка *Last-Modified*. Для получения одного бита данных s_3 повторно обращается к сущности e_3 и сравнивает новое значение HTTP-date$_1$ заголовка *Last-Modified* со значением HTTP-date$_0$. Если значения в заголовках не совпадают, то субъектом s_1 была отправлена 1, а иначе 0.
2) На основе заголовка *If-Modified-Since*. Субъект s_3 обращается к сущности e_3 и получает HTTP-ответ e_4, содержащий начальное значение HTTP-date$_0$ заголовка *Last-Modified*. Затем субъект s_3 повторно обращается к сущности e_3 с заголовком «If-Modified-Since: HTTP-date$_0$» и получает HTTP-ответ e_4. Если код полученного ответа равен 200, то сущность e_3 была изменена и субъект s_1 отправил 1. Если код ответа равен 304, то сущность e_3 не изменялась и субъект s_1 отправил 0.
3) На основе заголовка *If-Unmodified-Since*. Субъект s_3 обращается к сущности e_3 и получает HTTP-ответ e_4, содержащий начальное значение HTTP-date$_0$ заголовка *Last-Modified*. Затем субъект s_3 повторно обращается к сущности e_3 с заголовком «If-Unmodified-Since: HTTP-date$_0$» и получает HTTP-ответ e_4. Если код полученного ответа равен 412, то сущность e_3 была изменена и субъект s_1 отправил 1. Если код ответа равен 200, то сущность e_3 не изменялась и субъект s_1 отправил 0.

Стоит отметить, что скрытые каналы на основе заголовков *If-Modified-Since* и *If-Unmodified-Since* возможны даже в случае, если на веб-сервере запрещено выставление заголовка *Last-Modified* в HTTP-ответе.

Заголовок *Last-Modified* хранит время последней модификации ресурса с точностью до секунды и обновляется веб-сервером, как правило, раз в секунду, поэтому максимальная пропускная способность скрытых каналов по времени на основе сущностей HTTP-date составляет 1 бит/с.

С целью оценки пропускной способности, достижимой на практике в сети интернет, реализован скрытый канал по времени на основе заголовка *Last-Modified* на языке программирования C с использованием библиотеки sys/socket.h. Скрытый канал реализован для модели угроз M_1: на веб-сервере субъект-нарушитель изменял запра-

шиваемый клиентом ресурс в зависимости от передаваемого бита данных. Реализация данного скрытого канала в рамках модели M_2 не может увеличить пропускную способность, однако позволяет повысить точность передачи ввиду того, что нет необходимости в синхронизации субъектов-нарушителей, поскольку в этом случае веб-сервер контролируется нарушителем и поэтому он может передавать следующий бит данных при поступлении запроса к ресурсу.

Введём следующие определения для оценки экспериментальных данных. *Минимальный корректный префикс* — это наименьшее количество бит, полученных от начала передачи данных и до первой ошибки. *Средний и максимальный корректные префиксы* являются количествами бит, переданных подряд без ошибок, в среднем и в лучшем случаях соответственно. *Точность передачи* — процентное отношение числа правильно полученных бит к общему числу переданных бит.

Результаты тестирования данной реализации представлены в табл. 1.

Таблица 1

Пропускная способность канала по времени на основе заголовка Last-Modified в модели M_1 в сети интернет

Интервал запроса, с	Мин. префикс, бит	Средний префикс, бит	Макс. префикс, бит	Скорость передачи, бит/с	Точность передачи, %
2	3400	10145	22143	0,5	99,87
1	3200	8848	19712	1	99,82

Таким образом, максимальная теоретическая пропускная способность скрытых каналов по времени на основе сущностей HTTP-date достижима в сети интернет.

4. Скрытые каналы по времени на основе сущностей entity-tag

Идентификатор entity-tag формируется, как правило, на основе данных *inode* документа в файловой системе, его размера и времени последнего изменения сущности [10, 11]. Можно выделить следующие три варианта реализации скрытого канала по времени на основе сущостей entity-tag.

1) На основе заголовка *ETag*. Субъект s_3 обращается к сущности e_3 и получает HTTP-ответ e_4, содержащий начальное значение entity-tag$_0$ заголовка $ETag$. Для получения одного бита данных s_3 повторно обращается к сущности e_3 и сравнивает новое значение entity-tag$_1$ заголовка $ETag$ со значением entity-tag$_0$. Если значения в заголовках не совпадают, то субъектом s_1 была отправлена 1, а иначе 0.
2) На основе заголовка *If-Match*. Субъект s_3 обращается к сущности s_3 и получает HTTP-ответ e_4, содержащий начальное значение entity-tag$_0$ заголовка $ETag$. Затем субъект s_3 повторно обращается к сущности e_3 с заголовком «If-Match: entity-tag$_0$» и получает HTTP-ответ e_4. Если код полученного ответа равен 412, то сущность e_3 была изменена и субъект s_1 отправил 1. Если код ответа 200, то сущность e_3 не изменялась и субъект s_1 отправил 0.
3) На основе заголовка *If-None-Match*. Субъект s_3 обращается к сущности e_3 и получает HTTP-ответ e_4, содержащий начальное значение entity-tag$_0$ заголовка $ETag$. Затем субъект s_3 повторно обращается к сущности e_3 с заголовком «If-None-Match: entity-tag$_0$» и получает HTTP-ответ e_4. Если код полученного ответа равен 200, то сущность e_3 была изменена и субъект s_1 отправил 1. Если код ответа 304, то сущность e_3 не изменялась и субъект s_1 отправил 0.

Скрытые каналы на основе заголовков *If-Match* и *If-None-Match* также возможно реализовать, даже если веб-сервер не выставляет заголовок *ETag*.

Поскольку сущности entity-tag, как правило, содержат время последней модификации ресурса в формате UNIX-time (количество микросекунд с 01.01.1970), то теоретическая пропускная способность данных скрытых каналов по времени равна 10^6 бит/с. Стоит отметить, что частота обновления заголовка не специфицирована в RFC и составляет, как правило, 1 с.

С целью оценки пропускной способности, достижимой на практике в сети интернет, реализован скрытый канал по времени на основе заголовка *ETag* на языке программирования C с использованием библиотеки sys/socket.h. Скрытый канал реализован для двух рассматриваемых моделей угроз. В рамках модели M_1 для сети интернет получены результаты, представленные в табл. 2.

Таблица 2

Пропускная способность канала по времени на основе заголовка ETag в модели M_1 в сети интернет

Интервал запроса, с	Мин. префикс, бит	Средний префикс, бит	Макс. префикс, бит	Скорость передачи, бит/с	Точность передачи, %
1	3200	8848	19712	1	99,82
0,5	2400	8142	18123	2	99,5

Для того чтобы максимально эффективно использовать пропускную способность данных скрытых каналов, необходимо, чтобы значение заголовка обновлялось при каждом изменении веб-сущности. В этом случае максимальная пропускная способность для каналов, основанных на заголовках *ETag*, составляет 1 бит за $(L+T)$ секунд, где L — время, необходимое s_3 для выполнения запроса к e_3 и получения ответа e_4, а T — время, необходимое субъектам s_2 и s_3 для вычислительных операций (сравнения значений заголовков, чтения и записи битов и т. п). Это может быть реализовано в модели нарушителя M_2. Предположения данной модели нарушителя обусловлены необходимостью изменить частоту выставления сервером заголовка *ETag*, чтобы максимально эффективно использовать предоставляемую им точность хранения времени. Для этого нарушитель должен иметь одну из следующих возможностей:

— изменять конфигурацию веб-сервера, увеличив частоту автоматического обновления заголовка *ETag*;
— изменять значение заголовка *ETag* в ответе HTTP после его формирования веб-сервером;
— использовать интерфейс выставления заголовков HTTP в динамических веб-страницах, предоставляемый данным веб-сервером, например, функцию header() языка PHP.

Таким образом, нарушитель должен иметь контроль над веб-сервером. Возможен также вариант использования модели M_1, если реализуется скрытый канал с использованием файлового хостинга, данный вариант рассмотрен далее. Важно, что даже при самостоятельном формировании значения заголовка *ETag* используется тот же алгоритм формирования значения заголовка (*inode-size-mtime*). Следовательно, сформированные нарушителем значения заголовков неотличимы от оригинальных ничем, кроме частоты обновления.

Для оценки пропускной способности в рамках модели угроз M_2 использовалась аналогичная реализация клиентской программы с использованием библиотеки sys/socket.h языка C. На стороне сервера использовалось веб-приложение, реализующее описанную выше логику работы на языке PHP. Полученные результаты представлены в табл. 3.

Таблица 3

Пропускная способность канала по времени на основе заголовка ETag в модели M_2 в компьютерных сетях различного типа

Тип компьютерной сети	Среднее значение HTTP RTT, мс	Скорость, бит/с
Узел сети	0,55	986
ЛВС ЦОД «*DigitalOcean*»	1,63	845,65
ЛВС	6,9	295,69
Интернет	110,2	13,22

Определим время отклика (HTTP round-trip time — RTT) как время, необходимое субъекту s_3 для того, чтобы сделать запрос к e_3 и получить ответ e_4 по протоколу HTTP. Пропускная способность рассматриваемых скрытых каналов существенно зависит от данного параметра, поэтому тестирование реализаций проводилось в компьютерных сетях различного типа:

— узел сети: субъекты s_1 и s_3 находятся на одном узле и общаются по протоколу HTTP;
— ЛВС ЦОД: субъекты s_1 и s_3 находятся на различных узлах одного ЦОД (в экспериментах использовался один из облачных ЦОДов «Digital Ocean»);
— ЛВС: s_1 и s_3 находятся на различных узлах одной ЛВС;
— интернет: s_1 и s_3 находятся на различных узлах, подключенных к сети интернет.

Кроме того, наложенные ограничения позволили s_2 изменять время последней модификации e_3 при формировании ответа на запрос s_3. Таким образом, появилась возможность отказаться от синхронизации интервалов между запросами s_1 и s_3 и избежать потерь данных в случаях, когда значение HTTP RTT для конкретного запроса превышало используемый интервал между запросами. Данное решение позволяет гарантировать 100 %-ю точность скрытого канала.

5. Реализация скрытых каналов по времени на основе заголовков кэширования в веб-браузерах

Реализация рассматриваемых скрытых каналов в веб-браузерах представляет отдельный интерес, поскольку последние являются основным клиентским приложением, использующим протокол HTTP, и в то же время накладывают существенные ограничения на используемые интерфейсы сетевого взаимодействия. Например, имеющимися средствами веб-браузеров в настоящее время невозможно получить доступ к низкоуровневым структурам IP- и TCP-заголовков сетевых пакетов, а также к сообщениям HTTP вне контекста (origin) веб-сервера. Вместе с тем, получив возможность исполнять произвольные сценарии Javascript в веб-браузере пользователя, нарушитель может запустить DDoS-атаку в отношении некоторого целевого сервера, получить доступ к конфиденциальной информации, доступной через интерфейс веб-браузера, выполнить сканирование внутренней локальной сети и т. д. Следовательно, возникает задача реализации коммуникационного канала для передачи команд и данных между серверами управления нарушителя и веб-браузером пользователя. Одним из способов

реализации таких коммуникационных каналов могут быть скрытые каналы. Таким образом, основная цель реализации скрытых каналов в веб-браузерах — это затруднение обнаружения и анализа сетевыми средствами контроля и фильтрации передаваемых данных между вредоносным сценарием Javascript, запущенным в веб-браузере пользователя (hooked browser), и сервером управления злоумышленника [14].

В настоящее время известные скрытые каналы по времени, допускающие реализацию в веб-браузерах, основаны на использовании допустимого времени доступа (таймеров) или DNS-туннелирования. Скрытый канал по времени на основе допустимого времени доступа предложен в [9] и реализован для протокола HTTP в [8]. Данный подход может быть использован для всех протоколов. Его идея заключается в следующем: два субъекта s_1 и s_2 договариваются о максимально возможном времени доступа t. Если субъект s_1 до истечения времени t получает запрос от s_2, то это означает передачу бита 1; если он не получает запроса, то был передан бит 0. Пропускная способность данного скрытого канала для протокола HTTP в ЛВС составляет 1,82 бит/с [8].

Скрытые каналы по времени на основе DNS-туннелирования [14, 15] реализуются следующим образом: веб-браузер отправляет DNS-запросы к серверу и интерпретирует ответы на них как последовательность бит. Например, сообщение об ошибке типа NXdomain на запрос $bit1.evil.com$ клиент интерпретирует как бит 0, а ответ на запрос $bit2.evil.com$, содержащий некоторый IP-адрес, как бит 1. Пропускная способность скрытого канала на основе DNS в сети интернет составляет примерно 10 бит/с.

Реализация предложенного в работе семейства скрытых каналов на основе заголовков кэширования в веб-браузере основана на использовании средств DOM, Javascript и HTML, накладывающих следующие дополнительные ограничения:

— невозможность синхронной остановки работы алгоритма на определённое время;
— низкая точность существующих асинхронных механизмов управления временем;
— сложность синхронизации субъекта-получателя и субъекта-отправителя;
— низкая точность передачи.

Недостижимость высокой синхронности работы кооперирующихся субъектов-нарушителей делает реализацию исследуемых скрытых каналов по времени нецелесообразной в рамках модели угроз M_1. Однако отказ от синхронизации, как в модели угроз M_2, позволяет обойти налагаемые средой веб-браузера ограничения и получить практическую реализацию скрытых каналов на основе заголовков кеширования протокола HTTP в веб-браузере.

Для оценки достижимой на практике пропускной способности исследуемых скрытых каналов по времени в веб-браузере разработаны клиентский сценарий, написанный на языке Javascript, и два серверных сценария, написанных на языках PHP и Python и размещённых на веб-серверах Apache и Flask соответственно. Полученные результаты представлены в табл. 4.

Таблица 4

Пропускная способность скрытых каналов по времени в веб-браузере

Заголовок	Версия сервера	HTTP RTT, мс	Скорость, бит/с
Last-Modified	Python	70	1
Last-Modified	PHP	68	1
ETag	Python	66	11,51
ETag	PHP	72	10,8

6. Реализация скрытого канала по времени с использованием облачных файловых хранилищ данных

Основным недостатком модели угроз M_1 является низкая пропускная способность скрытого канала, реализованного в её ограничениях. Как уже было сказано, наиболее распространённой частотой обновления заголовков кэширования является 1 с, а отсутствие контроля над веб-сервером s_2 (в том числе и невозможность модификации сущности e_4) в модели M_1 не позволяет нарушителю увеличить частоту обновления заголовка, а значит, и пропускную способность канала. В то же время использование в качестве s_2 веб-сервера с большей частотой обновления заголовков кэширования позволяет решить проблему низкой пропускной способности.

В качестве таких серверов возможно использование веб-серверов облачных файловых хранилищ данных. Основной целью облачных хостингов является размещение информации на сервере и обеспечение к ней доступа из сети интернет. Повышенная частота обновления заголовков кэширования обусловлена необходимостью поддержания актуальной информации о размещённых в облаке файлах, даже если они меняются чаще, чем раз в секунду. Кроме того, большинство крупных облачных хранилищ, например Dropbox или Google Drive, предоставляют программный интерфейс (API) для управления уже размещенными на сервере файлами: их загрузки и выгрузки, систематизации и обновления метаинформации (в том числе и времени последнего изменения файла). Таким образом, используя API, предоставляемые облачными хранилищами, можно реализовать скрытый канал на основе заголовков кэширования в рамках модели M_1, обеспечивающий свойство анонимности — субъекты s_1 и s_3 имеют возможность обмениваться данными, не взаимодействуя друг с другом напрямую.

В качестве платформы для тестирования выбран облачный сервис Google Drive. На хостинге 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 совершает POST-запрос по адресу `https://www.googleapis.com/drive/v2/files/fileId/touch`, тем самым изменяя время последнего доступа к e_3 (здесь и далее fileId — идентификатор e_3 на облачном хостинге). Для получения переданного бита субъект s_3 получает значение используемого заголовка из метаинформации о файле, совершая GET-запрос по адресу `https://www.googleapis.com/drive/v2/files/fileId`, после чего восстанавливает переданный бит, руководствуясь описанными выше алгоритмами. Полученные данные представлены в табл. 5.

Таблица 5

Пропускная способность скрытого канала по времени с использованием Google Drive на основе *ETag*

Длина сообщения, бит	Точность, %	Скорость, бит/с	Ping RTT, мс
256	99,87	2,92	0,418
512	99,84	2,9	0,418
1024	99,8	2,88	0,418
2048	99,8	2,88	0,418
4096	99,87	2,86	0,418

Как видно из полученных результатов, пропускная способность скрытого канала в модели M_1 выросла и составляет теперь 1 бит за $(L + T + S)$ секунд, где L — время, необходимое s_3 для выполнения запроса к e_3; T — время, необходимое s_3 для вычислительных операций; S — время, уходящее на обработку запроса на серверах хостинга.

В данном случае время S выделяется отдельно, так как существенно зависит не от клиентов скрытого канала, а от выбранного для общения хостинга; при тестировании данное время составляло около 300 мс.

Таким образом, описанный скрытый канал сочетает в себе лучшие качества модели M_1, а именно: низкие требования; анонимность клиентов и трудность обнаружения и анализа сетевыми средствами контроля и фильтрации; кроме того, он обладает пропускной способностью, сравнимой с пропускной способностью скрытых каналов в модели угроз M_2. Высокий уровень доверия крупным облачным сервисам также играет на пользу данному скрытому каналу и затрудняет его обнаружение в защищённых средах.

7. Реализация скрытого канала по времени на основе заголовка *ETag* в BeEF

Browser Exploatation Framework (BeEF) — это инструментальное средство анализа защищённости информационных систем, ориентированное на использование веб-браузеров и эксплуатацию их уязвимостей [14].

Реализация скрытого канала по времени на основе заголовка *ETag* в BeEF состоит из двух частей. Первая часть — это расширение BeEF, написанное на языке Ruby и выполняющее функции специализированного веб-сервера, реализующего логику выставления заголовка *ETag* в соответствии с описанным выше методом. Вторая часть — это модуль BeEF, написанный на языке Javascript и реализующий клиентскую часть скрытого канала.

Исходный код реализованного скрытого канала доступен в репозитории BeEF [16] в следующих директориях:

— `/beef/extensions/etag` — серверная часть;
— `/beef/modules/ipec/etag_client` — клиентская часть.

При тестировании пропускной способности скрытого канала получены результаты, представленные в табл. 6.

Таблица 6

Пропускная способность скрытого канала по времени на основе *ETag* в BeEF

Тип сети	Скорость передачи 256 бит, бит/с	Скорость передачи 1024 бит, бит/с	Ping RTT, мс	HTTP RTT, мс
Local host	10,11	9,9	0,045	0,6
ЛВС «Digital Ocean»	10,08	9,84	0,062	0,83
ЛВС	10,03	9,78	18,046	19,8
Интернет	5,09	4,97	176,09	360,6

Заключение

В работе впервые описано семейство скрытых каналов по времени на основе заголовков кэширования протокола HTTP, обладающих следующими свойствами:

— для их реализации не требуется изменения грамматики сообщений HTTP;
— доступ к ресурсу на чтение, через который реализуется скрытый канал, не является блокирующим доступом и не приводит к возникновению ошибок передачи данных;
— возможна реализация анонимного скрытого канала, не требующего модификации веб-сервера;
— простота реализации;

— возможность реализации в веб-браузере;
— сложность обнаружения.

В рамках двух рассмотренных моделей угроз предложены способы реализации скрытых каналов по времени на основе заголовков кэширования в веб-браузерах, Google Drive API и BeEF. Описанные скрытые каналы превосходят по пропускной способности известные ранее скрытые каналы по времени [7, 8, 15] для веб-браузеров. Данная работа вошла в престижный топ-10 хакерских атак 2014 г. по версии компании «WhiteHat Security» [17].

ЛИТЕРАТУРА

1. *Lampson B. W.* A note on the confinement problem // Comm. ACM. 1973. No. 16(10). P. 613–615.
2. ГОСТ Р 53113.1 – 2008 Информационная технология. Защита информационных технологий и автоматизированных систем от угроз информационной безопасности, реализуемых с использованием скрытых каналов. Ч. 1. Общие положения.
3. ГОСТ Р 53113.2 – 2009 Информационная технология. Защита информационных технологий и автоматизированных систем от угроз информационной безопасности, реализуемых с использованием скрытых каналов. Ч. 2. Рекомендации по организации защиты информации, информационных технологий.
4. Timing Channels. `http://www.multicians.org/timing-chn.html`
5. CWE-514. Covert Channel. `https://cwe.mitre.org/data/definitions/514.html`
6. CWE-385. Covert Timing Channel. `https://cwe.mitre.org/data/definitions/385.html`
7. *Alkorn W., Frichot C., and Orru M.* The Browser Hacker's Handbook. Indianapolis: John & Wiley Sons, 2014. 648 p.
8. *Brown E., Yuan B., Johnson D., and Lutz P.* Covert channels in the HTTP network protocol: Channel characterization 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.
9. *Cabuk S., Brodley C.E., and Shield C.* IP covert timing channels: design and detection // Proc. 11th ACM Conf. on Computer and Communication Security. Washington DC, USA, 2004. P. 178–187.
10. RFC 2616. Hypertext Transfer Protocol HTTP 1.1. `http://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html`
11. Apache Core Features Documentation. FileETag Directive. `http://httpd.apache.org/docs/2.2/mod/core.html#fileetag`
12. *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.
13. Application Threat Modelling. `https://www.owasp.org/index.php/Application_Threat_Modeling`
14. The Browser Exploitation Framework Project. `http://beefproject.com/`
15. *Born K.* Browser-Based Covert Data Exfiltration. `http://arxiv.org/ftp/arxiv/papers/1004/1004.4357.pdf`
16. Исходный код BeEF. `https://github.com/beefproject/beef`
17. Top 10 Web Hacking Techniques of 2014. `https://blog.whitehatsec.com/top-10-web-hacking-techniques-of-2014/`

REFERENCES

1. *Lampson B. W.* A note on the confinement problem. Comm. ACM, 1973, no. 16(10), pp. 613–615.
2. GOST R 53113.1 – 2008 Informatsionnaya tekhnologiya. Zashchita informatsionnykh tekhnologiy i avtomatizirovannykh sistem ot ugroz informatsionnoy bezopasnosti, realizuemykh s ispol'zovaniem skrytykh kanalov. Part 1. Obshchie polozheniya [Information Technology. Protection of Information Technology and Automated Systems from Threats to Information Security Implemented using Covert Channels. Part 1. General Provisions]. (in Russian)
3. GOST R 53113.2 – 2009 Informatsionnaya tekhnologiya. Zashchita informatsionnykh tekhnologiy i avtomatizirovannykh sistem ot ugroz informatsionnoy bezopasnosti, realizuemykh s ispol'zovaniem skrytykh kanalov. Part 2. Rekomendatsii po organizatsii zashchity informatsii, informatsionnykh tekhnologiy. [Information Technology. Protection of Information Technology and Automated Systems from Threats to Information Security Implemented using Covert Channels. Part 2. Recommendations on the Organization of Information Security and Information Technology]. (in Russian)
4. Timing Channels. `http://www.multicians.org/timing-chn.html`
5. CWE-514. Covert Channel. `https://cwe.mitre.org/data/definitions/514.html`
6. CWE-385. Covert Timing Channel. `https://cwe.mitre.org/data/definitions/385.html`
7. *Alkorn W., Frichot C., and Orru M.* The Browser Hacker's Handbook. Indianapolis, John & Wiley Sons, 2014. 648 p.
8. *Brown E., Yuan B., Johnson D., and Lutz P.* Covert channels in the HTTP network protocol: Channel characterization 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, pp. 56–65.
9. *Cabuk S., Brodley C.E., and Shield C.* IP covert timing channels: design and detection. Proc. 11th ACM Conf. on Computer and Communication Security. Washington DC, USA, 2004, pp. 178–187.
10. RFC 2616. Hypertext Transfer Protocol HTTP 1.1. `http://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html`
11. Apache Core Features Documentation. FileETag Directive. `http://httpd.apache.org/docs/2.2/mod/core.html#fileetag`
12. *Devyanin P. N.* Modeli bezopasnosti komp'yuternykh sistem. Upravlenie dostupom i informatsionnymi potokami: ucheb. posobie dlya vuzov, 2-e izd., ispr. i dop. [Models of the Computer Systems Security. Access and Information Flow Control]. Moscow, Goryachaya Liniya-Telekom Publ., 2013. 338 p. (in Russian)
13. Application Threat Modelling. `https://www.owasp.org/index.php/Application_Threat_Modeling`
14. The Browser Exploitation Framework Project. `http://beefproject.com/`
15. *Born K.* Browser-Based Covert Data Exfiltration. `http://arxiv.org/ftp/arxiv/papers/1004/1004.4357.pdf`
16. Source Code BeEF. `https://github.com/beefproject/beef`
17. Top 10 Web Hacking Techniques of 2014. `https://blog.whitehatsec.com/top-10-web-hacking-techniques-of-2014/`