

# Digital Security Research Group

Результаты исследования  
безопасности банк-клиентов  
российских производителей  
за период 2009-2011 гг.

«Правило первое: никогда не теряй деньги.  
Правило второе: никогда не забывай первое правило»

Уоррен Эдвард Баффетт

Исследовательский центр Digital Security Research Group (<http://www.dsecrg.ru>), открытый компанией Digital Security в 2007 году, практически с момента своего основания проводит регулярные ежегодные исследования безопасности отечественных систем ДБО. Это связано с повышенной критичностью ПО данного класса, интересом к этому вопросу со стороны злоумышленников, а также с отсутствием какой-либо информации о безопасности таких продуктов.

## Цели исследования

Цель исследования — оценить отечественный рынок систем ДБО с точки зрения безопасности, обратив внимание на выявление ошибок в коде, приводящих к появлению уязвимостей и ошибок логики, а также на использование защитных механизмов для снижения риска атак. Конечная цель — выявить возможные векторы атак на системы ДБО без доступа к рабочей станции клиента банка сторонними средствами. Таким образом, в данной работе предполагается, что клиенты банка не заражены никаким вредоносным ПО, и предмет исследования ограничен проблемами безопасности ПО систем ДБО, не затрагивая проблем браузера, ОС или иного стороннего ПО.

## Исследуемые системы

В данном заключении представлены результаты тестирования и исследования различных систем следующих производителей:

- БСС;
- Р-Стайл;
- CompassPlus;
- Сигнал КОМ;
- ЦФТ;
- ИНИСТ;
- Степ Ап.

В рамках данной работы не будут приводиться технические детали существующих проблем или указания на конкретные продукты, с целью предотвращения воспроизведения проблемы на работающих и доступных системах.

## Архитектура

Существует множество различных видов дистанционного обслуживания различных клиентов банка. От этого зависит тип ПО, архитектура системы и многое другое. Мы разделяем системы ДБО по принципу доступа к интерфейсу и по принципу работы с системой. Так, существуют:

- «интернет-клиент» — система с доступом только с помощью браузера;
- «тонкий клиент» — система, аналогичная интернет-клиенту, отличающаяся тем, что в браузер встраивается дополнительная надстройка для работы с ключами клиента и установки электронной цифровой подписи (ЭЦП) на платежные поручения. Такой механизм чаще используется для клиентов — юридических лиц;
- «толстый клиент» — система, для работы с которой используется отдельное (от браузера) ПО, запускаемое клиентом при организации доступа.

## Используемая защита

Защита перечисленных систем сводится к использованию идентификатора пользователя и пароля, а также второго фактора аутентификации. Кроме того, существуют каналы оповещения пользователя о событиях в системе (о снятии денег со счета, например). Иные технические защитные средства зависят от заказчика (банка).

## Программные ошибки

Большинство систем было разработано на популярных языках программирования, при этом компоненты систем могут быть как веб-приложениями (например, технологии ASP.Net, Java), так и клиентскими приложениями (технология ActiveX) или просто приложениями, написанными на языках семейства Си. В совокупности все эти компоненты обеспечивают функционал для обеспечения удаленного доступа клиента к своим счетам в банке. Используя данную базу для создания таких систем, программисты могут совершать ошибки, которые приводят к появлению уязвимостей как на клиентской части, так и на серверной.

Основные уязвимости при разработке толстого клиента, ActiveX-компонентов, а также серверной части, если это не веб-приложение, появляются в результате ошибок памяти и реализуются через ошибки при управлении переменными в стеке, куче или при неверной работе с указателями, счетчиками или индексами. Класс таких ошибок довольно широк и включает в себя такие известные уязвимости, как:

- переполнение буфера в стеке/куче;
- использование памяти после освобождения;
- целочисленное переполнение.

В худшем случае, такие уязвимости приводят к угрозе выполнения произвольного кода на хосте с правами учетной записи, из-под которой запущено атакуемое ПО системы ДБО. В иных случаях это может приводить к отказу в обслуживании, так как атакуемый процесс аварийно завершает свою работу. В случае, если мы говорим о клиентском ПО (ActiveX), наибольшую опасность представляет собой только риск произвольного выполнения кода, так как потенциальный нарушитель сможет выполнить захват рабочей станции клиента банка. В случае серверного ПО риск отказа в обслуживании также считается критичным, так как может принести большой убыток банку за счет остановки бизнес-процессов. Также существует вероятность, что атаки типа отказа в обслуживании могут использоваться злоумышленником для отвлечения внимания (как сотрудников банка, так и клиента, который не может в такой ситуации проверить состояние счета) от своих действий, которые могли быть совершены до DoS-атаки.

*Наши исследования показали, что все описанные виды уязвимостей присутствуют в ПО систем ДБО. Кроме того, в той или иной степени все исследуемые продукты содержали те или иные из перечисленных уязвимостей, что говорит об общем уровне качества кода с точки зрения безопасности.*

Так как некоторые уязвимости были достаточно очевидными, обнаруживались методом модификации легитимных запросов и даже находились в «предсказуемых» местах, можно сказать, что проблема является комплексной и связана не столько с ошибкой конкретного программиста в конкретной компании-разработчике, сколько с общим отсутствием контроля качества безопасности кода при разработке. Это означает, что нет таких процессов, как:

- проверка кода;
- стресс-тестирование (фаззинг);
- анализ безопасности на уровне архитектуры.

Все это подтверждается и тем фактом, что некоторые уязвимости (веб) находились по доступными сходным кодам веб-приложений (бесплатными средствами поиска уязвимостей по тексту кода). Другими словами, даже та часть задачи обеспечения безопасности кода, которая решается использованием автоматизированных средств, игнорируется разработчиками.

Наличие различных уязвимостей, обнаруженных в ПО веб-сервисов систем ДБО (всего: 6):

- Межсайтовый скриптинг — 6 продуктов
- Инъекция SQL-кода — 3 продукта
- Ошибки авторизации — 4 продукта
- Межсайтовые запросы — 5 продуктов

При этом наша статистика за три года исследования ДБО показывает, что общее количество и уровень опасности уязвимостей остаются неизменными. Все виды уязвимостей каждый год (2009, 2010, 2011) обнаруживались в различных продуктах.

Эта небольшая статистика лишь подтверждает, что с точки зрения безопасности кода все компании-производители действуют примерно на одном общем уровне. Тем не менее, некоторые компании из списка реагируют на проблемы лучше, чем иные (выделяться они не будут, чтобы избежать рекламы).

Ситуация с клиентским ПО (ActiveX) аналогична: уязвимости типа «выполнение произвольного кода» и вызов «небезопасных методов» были найдены в 4 из 6 систем.

*Сам факт наличия уязвимостей в ПО и то, что по прошествии трех лет они по-прежнему существуют, говорит о том, что в целом разработчики не изменили свое отношение к производству продукта.*

В качестве показательного примера давайте рассмотрим одного из производителей, ПО которого мы анализировали в 2010 году методом фаззинга.

В итоге была найдена уязвимость (в серверной части ПО) типа «переполнение буфера в стеке». Уязвимость была исправлена через два дня. Ровно через год, в 2011, мы повторили фаззинг того же протокола и нашли похожую проблему – целочисленное переполнение. Другими словами, разработчик в 2010 году получил информацию об ошибке в коде. Исправил ее, но проводить собственное исследование с целью поиска ошибок не стал, таким образом «сохранив» уязвимость на следующий год.

То же было и с другими производителями. Так, сообщив в 2009 году об ошибках в ActiveX-компонентах системы, в 2011 мы нашли похожие ошибки в другой функциональной части того же модуля. Это также является косвенным доказательством того, что прогресса с точки зрения изменения процессов обеспечения безопасности ПО в компаниях-производителях за последнее время не произошло.

## Распространение обновлений

Помимо анализа обнаружения ошибок во временной перспективе, также было отслежено, как производитель распространяет обновления для своих систем среди банков (клиентов). Результат довольно любопытный: все те ошибки, что мы обнаружили за период 2009–2010, можно встретить и в 2011 у различных банков, которые не обновили ПО системы по каким-либо причинам. Такие факты были выявлены неоднократно.

*Более того, было выяснено, что банки вообще не в курсе, что в их ПО существуют проблемы с безопасностью, несмотря на то, что разработчик этого ПО знает об ошибках более года и даже выпустил обновления, решающие эти проблемы.*

Это позволяет сделать вывод о наличии проблем коммуникации между производителем и клиентом на различных уровнях (проблемы в менеджменте – много разных банков; проблемы в маркетинге – удар по репутации; технические проблемы – сложность создания исправления, подходящего для всех клиентов).

## Технологический уровень защиты

При разработке ПО — как веб-проектов, так и бинарных приложений — существуют механизмы и технологии снижения рисков при реализации различных типовых уязвимостей и атак. В результате исследования систем ДБО было выявлено, что в большинстве продуктов не применяется большая часть защитных механизмов.

Защитные механизмы web:

- Флаги защиты cookie: HTTPOnly, Secure;
- Поддержка SSL при аутентификации и доступе к системе;
- Контроль загрузки во фрейме (FrameBusting);
- Токены запросов.

Защитные механизмы бинарных приложений:

- Защитная метка от переполнения буфера в стеке;
- Поддержка работы модуля с использованием флага NX (DEP);
- Поддержка работы модуля со случайными базовыми адресами (ASLR);
- Поддержка защиты от атак на структуры обработки исключительных ситуаций (SafeSEH).

В случае с веб-приложениями, лишь один разработчик применил уникальные токены запросов, которые защищают от атак класса «межсайтовые запросы» (CSRF). Однако почти все системы не поддерживают методики защиты типа FrameBusting и использование флагов cookie, которые снижают риски от атак типа «межсайтовый скриптинг» (XSS). Бинарные модули также не поддерживают современные защитные технологии. Лишь один продукт был скомпилирован с флагом /GS (предупреждение от атаки типа «переполнение буфера в стеке»). Эти факты лишь подтверждают предыдущие выводы о процессе разработке кода и фактически говорят о том, что современные хакерские атаки на ПО без проблем могут быть реализованы для отечественных систем ДБО.

## Работа с криптомодулями

Важной чертой отечественных систем ДБО является использование ключей для установки ЭЦП на документы перед их отправкой на сервер. При этом хорошей практикой является использованием клиентом токена с неизвлекаемым ключом. Мы не касаемся в данной работе возможности отправки поддельного платежного поручения при компрометации ПК клиента банка (например, с помощью трояна), поэтому рассмотрим архитектуру работы ДБО с ЭЦП с точки зрения возможности отправки поддельных платежных поручений при невозможности влиять на ПК клиента (доверенная среда пользователя обеспечена).

На практике нам удалось реализовать атаку на ПК клиента через уязвимости в веб-модулях ДБО (XSS, CSRF). В совокупности с отсутствием защитных механизмов, описанных выше, такая атака приводит к подделке платежного поручения с валидной ЭЦП пользователя. Схема атаки:

1. Через уязвимость межсайтового скриптинга подгружается фрейм со страницей ДБО, развернутой на весь экран.
2. Скрипт атакующего в основном окне контролирует все отображаемое содержимое во фрейме. При этом пользователь видит только фрейм с интерфейсом ПО.
3. Как только пользователь начинает операции со счетом, в частности создание платежного поручения, то скрипт атакующего подменяет реквизиты получателя на реквизиты атакующего. При этом в интерфейсе отображается информация, введенная пользователем.
4. В момент подписи платежного поручения в токен отправляется поддельное платежное поручение. На экране отображается платежное поручение пользователя.
5. Подписанное платежное поручение атакующего отправляется пользователем с валидной ЭЦП. На экране компьютера пользователь видит те данные, которые он ввел самостоятельно.

Данная атака известна как «человек в браузере» (MitB). Однако в данном примере ПК пользователя не заражен вредоносным ПО, и атакующий выполняет атаку, используя исключительно ошибки системы ДБО. Более того, системы с двумя подписями также уязвимы к данной атаке, так как обычно с ДБО работает специальный оператор, который имеет оба токена. При этом выход из системы для установки второй подписи не мешает атаке, так как фрейм по-прежнему контролируется родительским окном, а значит и злоумышленником. Мы тестировали данную атаку не только технически, но и практически – на операторах, которые работают с такими системами. В итоге они отправляли тестовые поддельные переводы (на тестовом стенде), не замечая атаки и подделки реквизитов получателя, что говорит о потенциале такого рода угроз.

Для нескольких продуктов нам удалось реализовать еще более опасный вид описанной выше атаки. Общий подход и используемые уязвимости те же (CSRF, XSS), но при этом от пользователя не требуется выполнять какие-то действия в системе ДБО. Используя JavaScript, запускаемый в скрытом для пользователя фрейме, можно имитировать его действия и отправить подписанное платежное поручение. Таким образом, злоумышленник, заманив клиента ДБО на специальный сайт, имеет возможность тайно украсть его деньги.

Кроме того, были замечены общие подходы при реализации работы токена с веб-системой ДБО. Так, во многих системах ПИН-код от токена вводится средствами JavaScript/HTML (используя, например, методы ActiveX), и подпись любого документа осуществляется аналогичным образом. При этом если ПИН код уже был введен, то подпись документа происходит в невидимом режиме. Все эти обстоятельства сказываются на безопасности. Например, атакующий может взаимодействовать с ActiveX-компонентом токена или системы ДБО, выполняя скрытый перебор ПИН-кодов с вредоносного сайта, и оттуда же может выполнить подпись любых данных, в том числе и платежного поручения. В совокупности с уязвимостями класса XSS на серверах ДБО или даже домена банка такие архитектурные подходы очень сильно повышают риск успешной атаки.

Некоторые исследуемые системы использовали систему прозрачного шифрования и подписи средствами локального прокси-сервера – специального ПО, которое работает с токеном. При такой архитектуре данное ПО выполняет установку подписи «на лету», например, при определенных параметрах HTTP-запросов. Слабые места такой архитектуры очевидны: в случае генерации неавторизованных HTTP-запросов (как при CSRF-атаке) подпись будет установлена автоматически.

## Внутренние архитектурные проблемы

Кроме вышеперечисленных недостатков в ряде отечественных систем ДБО были выявлены глобальные ошибки архитектуры. Так, в некоторых системах ДБО отсутствуют повторные проверки ЭЦП на платежных поручениях при выгрузке их в АБС (Автоматизированная Банковская Система). Очевидно, что фактически операции со счетами происходят в АБС, тогда как в системе интернет-банкинга происходят лишь «указания» для АБС, которая об ЭЦП ничего не знает. Таким образом, получается, что пользователь взаимодействует с системой интернет-банкинга, которая и проверяет ЭЦП, а затем экспортирует платежные поручения в БД АБС. Далее оператор банка обрабатывает платежное поручение в АБС. Разработчики системы ДБО понимают, что АБС недоступна из сети Интернет и проверку ЭЦП достаточно делать только при приеме платежного поручения. Однако известные уязвимости и атаки (например, SQL-инъекция) могут скомпрометировать целостность результата этой проверки и тем самым форсировать выгрузку платежных поручений без ЭЦП в АБС, после чего оператор банка обработает их как нормальные (ведь в АБС не может быть ошибочных документов: они просто не выгрузятся с нужным статусом). Пример такой атаки – SQL-инъекция. Упрощенный алгоритм работы выглядит так:

1. Прием платежного поручения (ПП).
2. Выгрузка ПП в базу данных (БД) ДБО со статусом «без подписи».
3. Подпись данных ПП.
4. Проверка ЭЦП ПП открытым ключом клиента с помощью специального ПО.
5. Изменение статуса ПП в БД ДБО на «подпись верна».
6. Подтверждение клиентом отправки ПП.
7. Изменение статуса ПП в БД ДБО на «отправлено».

8. Специальный скрипт или ПО по событию или таймеру выгружает все ПП со статусом «отправлено» из БД ДБО в БД АБС.
9. Оператор, работая с АБС, обрабатывает ПП.

Атака с применением внедрения SQL-кода, в случае соответствующей уязвимости в ПО системы ДБО (в веб-интерфейсе), позволит выполнить шаг 5 без выполнения шага 4, что не нарушит логического хода алгоритма.

*Если говорить о работе внутри корпоративной сети банка, то системы ДБО предоставляют множество функций и возможностей для клиентов (сотрудников банка), но также и уязвимостей.*

В результате наших исследований мы выявили общие архитектурные проблемы при доступе к системе ДБО сотрудников банка. Очевидно, что все сотрудники при таком доступе проходят аутентификацию и даже могут разделяться по ролям. Однако, исследуя ПО, мы выяснили, что фактически доступ к СУБД происходит под одной ролью, имеющей доступ ко всей СУБД, а разделение по ролям ограничивает лишь само приложение клиента, установленное на рабочей станции пользователя. Фактически такая «двухзвенная» архитектура при хранении «прошитога» и зашифрованного общего пароля вообще не является защитой, ведь любой пользователь имеет возможность «вытащить» расшифрованный общий пароль из памяти клиентского процесса в момент расшифровки, что позволит повысить привилегии в СУБД.

Итак, на основании проведенных нами исследований и анализа ПОв нескольких продуктах обнаружилась возможность проводить произвольные платежные поручения, возможность вызова отказа в обслуживании серверов, возможность выполнения произвольного кода на серверах ДБО.

Для банков любой инцидент – это прежде всего потенциальный удар по репутации. Для клиентов банков проблемы с безопасностью ДБО Банка могут привести к более тяжким последствиям. Во-первых, это, конечно, кража денег. Причем в некоторых случаях, при успешной атаке на сервера ДБО, можно украсть деньги у любого из клиентов банка. Во-вторых, во всех проанализированных нами продуктах злоумышленник мог получить полную информацию о производимых клиентом операциях.

## Выводы

*В результате исследований мы убедились, что уровень безопасности современных отечественных систем ДБО находится на уровне 90-х годов.*

Не используются современные технологии при разработке ПО. Процессы разработки безопасного кода и архитектуры отсутствуют. Количество уязвимостей стабильно высоко на протяжении 3 лет по всему рынку. Некоторые уязвимости настолько просты и легко эксплуатируемы, что угроза очень высока (так, например, одна система отдавала полные номера банковских карт (PAN) при неудачной попытке аутентификации по существующему логину). Большинство уязвимостей приводит к раскрытию персональных данных, банковской тайны, а также нарушению целостности информации о счетах (подделка платежных поручений). Уровень зрелости с точки зрения ИБ отстает от аналогичных продуктов западного производства, хотя можно найти и сходства: чем менее популярно ПО на открытом рынке и более узко специализировано, тем меньше разработчики уделяют внимание ИБ, независимо от критичности продукта. Множество угроз остается за пределами внимания специалистов ИБ банков и разработчиков ДБО, тогда как о защите от них стоит позаботиться уже на стадии архитектуры и разработки.



№ 1 в аудите безопасности

Digital Security — одна из ведущих российских консалтинговых компаний в области информационной безопасности, предоставляющая полный спектр услуг, в том числе, проведение аудитов ИБ и тестов на проникновение, подготовку и сертификацию по PCI и PA-DSS, СТО БР ИББС, аудит защищенности систем ДБО, SCADA, ERP- и бизнес-приложений.

Являясь ведущей в СНГ компанией, специализирующейся на аудите защищенности систем ДБО, Digital Security предоставляет своим заказчикам высочайший уровень экспертизы в этой области.

В основе опыта компании лежит исследовательская деятельность подразделения Digital Security Research Group (DSecRG), занимающегося поиском и анализом уязвимостей в критичных для бизнеса приложениях таких компаний, как Oracle, SAP, 1С и прочие. С 2007 г. одним из приоритетных направлений исследований DSecRG является поиск уязвимостей в банк-клиентах ведущих российских производителей, в продуктах которых за это время было обнаружено множество критичных уязвимостей. Кроме того, за свою исследовательскую деятельность специалистами DSecRG было получено множество благодарностей от ведущих мировых вендоров, включая IBM, VMWare, SAP и Oracle и др.

Digital Security является разработчиком ERPScan сканера безопасности SAP — продукта по комплексной оценке защищенности и проверке соответствия стандартам для платформы SAP.

Специалисты DSecRG возглавляют международный проект OWASP-EAS по анализу защищенности бизнес-приложений и являются докладчиками на ведущих международных конференциях по практической безопасности, таких как BlackHat и HackInTheBox и многих других.

С 2009 года Digital Security ежегодно проводит конференцию PCI DSS Russia, а с 2011 года — международную хакерскую конференцию ZeroNights.

<http://www.dsec.ru>  
<http://www.dsecrg.ru>  
<http://www.pcidss.ru>

Тел.: +7 (812) 703-15-47