Между слоями осуществляется обмен данными. Правильное разделение приложения на слои помогает поддерживать строгое разделение функциональности, что в, свою очередь, обеспечивает гибкость, а также удобство и простоту обслуживания.

Отметим, что слои в клиентском приложении могут размещаться физически на одном компьютере (на одном уровне) или же быть распределены по разным компьютерам (n-уровней). Связь между компонентами разных уровней осуществляется через строго определенные интерфейсы.

На рис. 1 приведены основные логические слои архитектуры разработанного клиента.

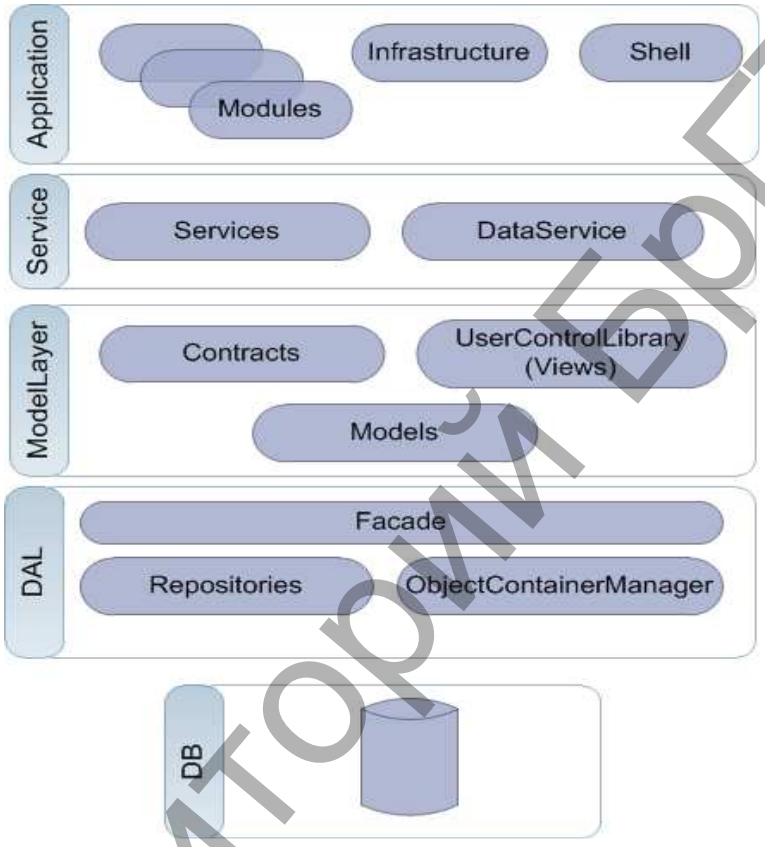

Рисунок 1 – Многоуровневая архитектура универсального клиента

Список цитированных источников

1. Нейгел, К. F# 2008 и платформа .NET 3.5 для профессионалов / К. Нейгел, Б. Ивьен, Дж. Глинн, К. Уотсон, М. Скиннер.: пер. с англ. – М.: ООО «И.Д. Вильямс» 2009. – 1392 с.

2. Рудикова, Л.В. Использование средств PowerDesigner для поддержки задач проектирования // Управление в социальных и экономических системах: материалы XV междунар. науч.-практ. конф. – Мн.: 2006. – С. 211–212.

УДК 004.056.5

УСИЛЕНИЕ БЕЗОПАСНОСТИ САЙТОВ ПОСРЕДСТВОМ НАСТРОЙКИ СЕТЕВЫХ СЕРВИСОВ И МИНИМИЗАЦИИ ОШИБОК ПРОГРАММИРОВАНИЯ

Чепонас А.С., Савельева Н.В.

УО «Витебский государственный университет им. П.М. Машерова», г. Витебск

На сегодняшний день в сети Интернет сайтами обладают не только крупные предприятия и корпорации, но даже и мелкие фирмы. Но если при этом успешные и развитые

организации могут позволить себе покупку качественного веб-приложения, то малый и средний бизнес на разработку и размещение своих сайтов не готов тратить большие суммы. Поэтому зачастую заказами на разработку сайтов занимаются студенты или школьники, а не группа квалифицированных веб-разработчиков. Более того, очень часто от создателя сайта требуется и размещение разработанного им продукта в Интернет или в локальной сети, т.е. на программиста налагаются задачи администрирования сервера. Очевидно, что в данном случае неизбежны многочисленные ошибки разного рода.

Вместе с тем наиболее популярным веб-сервером в Интернет является Apache, который обычно используется в т.н. связке LAMP (Linux, Apache, MySQL, PHP). Но несмотря на качественность компонентов LAMP, случаи успешного применения PHP-включений, SQL-инъекций, межсайтового скриптинга (XSS) и других атак, направленных на веб-сервисы, – далеко не редкость. Среди основных причин успешности таких атак называют широкую доступность инструментов, необходимых для проведения атаки, и недостаточное внимание со стороны разработчиков сайтов к вопросам безопасности. При этом условно можно выделить два основных фактора, снижающих безопасность: ошибки в администрировании сервера и ошибки в программировании веб-ресурса. Но говоря о защите информации, дополнительно можно остановиться еще и на защите самих данных, предоставляемых пользователю для отображения в браузере или для загрузки с сервера. Основная цель настоящей работы – систематизировать основные рекомендации по усилению безопасности веб-ресурсов, исходя из типичных ошибок программирования и администрирования.

Следующая таблица содержит основные рекомендации по безопасности сервера, приложения и данных.

1) Защита сервера	
– общие рекомендации по настройке серверного ПО	– своевременное обновление ядра и служб; – отключение неиспользуемых служб; – продуманная настройка правил межсетевого экрана; – логгирование системных событий и событий межсетевого экрана; – ограничение удаленного доступа к серверу.
– Apache [1]	– запрет отображения названия операционной системы, версии и информации об установленных модулях веб-сервера; – запуск веб-сервера под отдельной учетной записью в окружении chroot; – запрет чтения конфигурационных и временных файлов Apache; – ограничение доступа к определённым директориям или файлам; – отключение неиспользуемых модулей Apache; – использование digest-аутентификации; – ограничение выполнения CGI-скриптов, SSI-включений, индексирования каталога и следование символическим ссылкам; – при возможности ограничение доступа к веб-сайту по IP-адресу (диапазону IP-адресов) и/или по имени пользователя; – установка Timeout ожидания сервером ответа клиента; – для противостояния атакам отказа в обслуживании рекомендуется ограничить размер клиентского запроса, количество подключенных клиентов и пр. параметры, а также использовать дополнительный модуаль Apache – mod_evasive.
– php.ini [2]	– **register_globals=Off** – отключить глобализацию переменных; – **safe_mode=On** – включить жесткий режим ограничений;

	– **open_basedir= /www** – ограничить место выполнения PHP-кода; – **magic_quotes=On** – включить «магические кавычки» для GET/POST/COOKIE (с целью препятствия SQL-инъекциям); – **mysql.trace_mode=Off** – отключить показ ошибок MySQL; – **allow_url_fopen=Off** – отключить удаленное открытие файлов файловыми функциями; – **allow_url_include=Off** – отключить удаленное подключение файлов; – **error_reporting=Off** – отключить показ всех ошибок; – **disable_functions=exec, system, passthru** – задать ограничение на использование потенциально опасных функций; – не рекомендуется использовать параметры **default_host**, **default_user** и **default_password** секции **[MySQL]**.
– mysqld [3]	– после установки службы – изменение паролей всех пользователей; – запрет анонимного доступа к БД; – удаление тестовых таблиц и баз данных; – запуск службы в окружении chroot под отдельной учетной записью (не root), представляющей единственного пользователя, имеющего привилегии чтения/записи в директории БД; – разграничение ролей пользователей БД MySQL – в частности, пользователей, имеющих права на модификацию данных (INSERT, UPDATE, DELETE) и только на выборку (SELECT); – создание для каждой БД отдельного пользователя, который не имеет прав доступа к другим БД; – не создавать пользователей, имеющих привилегии на работу со структурой БД или административных привилегий; – не предоставлять привилегии PROCESS, FILE всем пользователям; – ограничение числа подключенных к БД пользователей; – если Web и MySQL сервера работают на одном компьютере, то рекомендуется «заставить» MySQL слушать только интерфейс локальной петли 127.0.0.1; – с целью противостояния атакам ботов рекомендуется изменить стандартный порт соединения с MySQL сервером.
	2) Защита приложения (сайта)
– общие рекомендации	– хранение паролей только в зашифрованном виде; – минимизация использования стороннего ПО для администрирования сайта; – программное ограничение скорости загрузки файлов с сервера (при необходимости); – логгирование событий приложения; – тестирование приложения перед опубликованием в сети.
– проверка пользовательского ввода (в т.ч. upload файлов)	– проверка вводимых пользователем данных на ожидаемые длину и тип; – проверка файлов, загружаемых на сервер, на тип и размер; – экранирование спецсимволов.
– сеансовые ключи и сеансы	– не передавать идентификатор сеанса через адресную строку; – генерация идентификатора сеанса достаточной длины и его шифрование; – установка таймаута жизни сеанса и привязка сеанса к IP-адресу; – задание собственного каталога для хранения файлов сеансов.
– Web IDS [4]	– рекомендуется использовать библиотеку PHPIDS (PHP Intrusion Detection System) для отслеживания разных видов атак – межсайтового скриптинга (XSS), SQL-инъекций, расщеплений запросов (HTTP Response Splitting), проходов по каталогам (Directory traversing), RFE/LFI, DoS, LDAP-инъекций и пр.

3) Защита данных	
– Общие рекомендации	– соблюдение мер противостояния SQL-инъекциям, указанных выше; – создание резервных копий и хранение их на отдельных накопителях; – обеспечение защиты авторского права доступных для загрузки файлов (например, защиту загружаемого с сервера файла паролем на открытие / редактирование/распечатывание, подписывание файла цифровой подписью, внедрение цифровых водяных знаков и пр.); – использование файла robots.txt для ограничения роботам доступа к содержимому веб-сайта.

Разумеется, приведенный в таблице список рекомендаций по мерам безопасности далеко не полон и не отражает частных деталей, однако является полезным ориентиром при внедрении и сопровождении информационных систем, использующих LAMP. В заключение отметим, что проектирование системы безопасности требует комплексного подхода, а обеспечение безопасности любого ресурса – не разовое мероприятие, а непрерывный многогранный процесс.

Список цитированных источников

1. Яремчук, С. Возьми индейца под защиту / С. Яремчук // Хакер, 2007. – №10(106). – С. 154.

2. Бойцев, О. Взлом и защита веб-сервера – на каждый яд есть свое противоядие / О. Бойцев // Компьютерная газета. – 2010. – №5.

3. Матвеев, А. На лезвии ножа / А. Матвеев // Хакер, 2005. – №3(75). – С. 52.

4. Зобнин, Е. Остаться на плаву. Обвески для Web-сервера, без которых не обойтись / Е. Зобнин // Хакер, 2010. – №10(133). – С. 128.

УДК 004.514.62

ПРОГРАММНАЯ СИСТЕМА ТЕСТИРОВАНИЯ ЭФФЕКТИВНОСТИ ОКОННЫХ ИНТЕРФЕЙСОВ

Шитиков А.В.

УО «Брестский государственный технический университет», г. Брест

Научный руководитель – Костюк Д.А., к.т.н., доцент

До самого недавнего времени оконные интерфейсы всех популярных операционных систем в той или иной мере были построены на основе метафоры «рабочего стола», получившей первое коммерческое воплощение во времена компьютеров Apple Macintosh. Согласно этой устоявшейся парадигме, существует основное (т. н. «корневое») окно графической среды, которое может полностью или частично перекрываться остальными окнами. Корневое окно отображает фоновое изображение, ряд основных элементов управления графической оболочки (обычно – средства переключения фокуса окон и запуска приложений), и, как правило, один из каталогов файловой системы, содержимое которого размещается в произвольном порядке поверх фонового изображения, по аналогии с предметами, лежащими на письменном столе.

Метафора рабочего стола на сегодняшний день воплощает многолетний опыт разработки графических интерфейсов, ориентированных на управление с помощью мыши. Однако интерфейсы на основе емкостных сенсорных экранов, управляемых пальцами,