21. *Трофимов С.* CASE-технологии: практическая работа в Rational Rose. – М.: Бином, 2002.

22. *Рубцов С.* Сравнительный анализ известных инструментов организационного проектирования. – http://www.vernikov.ru/content/view/461/126/.

УДК 004.056.53

А.А. Бондарев, А.К. Чернышов

СОЗДАНИЕ СИСТЕМЫ РАЗГРАНИЧЕНИЯ ДОСТУПА И ИЗОЛЯЦИИ СМЕЖНЫХ ДОЧЕРНИХ ПРОЦЕССОВ HTTP СЕРВЕРА APACHE С МОДУЛЕМ ПРЕВЕНТИВНОЙ ЗАЩИТЫ

С каждым годом количество информационных потоков увеличивается в геометрической прогрессии в глобальном масштабе. На фоне этого процесса, вопрос своевременного доступа к актуальной информации приобретает высший приоритет. В первую очередь это актуально для тех сфер, где объект информационного обмена не имеет физической природы (биржевой рынок, Интернет-магазины и т.д.).

Использование сетевых технологий даёт значительное преимущество в сравнении с более традиционными методами распространения информации. Единственной глобальной сетью, являющей собой глобальное информационное пространство, является веб.

Безусловно, самыми распространенными являются интерфейсы, работающие по протоколу HTTP. Однако статичное отображение данных не соответствует потребностям сегодняшнего дня. Перспективный веб-сервис, предоставляющий востребованную услугу, это, как правило, сложная система, состоящая из HTTP-сервера, системы управления базами данных и платформы для обработки скриптов на стороне сервера, результат выполнения которых является различным в зависимости от запросов пользователя.

В силу необходимости предоставления постоянного доступа извне экономически целесообразно использовать веб-сервер также для организации ряда дополнительных служб. На практике наиболее часто встречаются: корпоративный почтовый сервер, корпоративный сервер баз данных, службы межсетевого экранирования, службы биллинга и подсчета трафика. В подобных случаях веб-сервер является наиболее критичным звеном в инфраструктуре организации и наиболее вероятной целью потенциального злоумышленника.

Современный веб-сервер – это критически важная система, которая играет важную роль в инфраструктуре организации, потенциально содержащая конфиденциальные данные. Поиск возможных решений для ряда проблемных вопросов затрагивающих вопросы использования ресурсов одного сервера несколькими сайтами на сегодняшний день крайне актуален и востребован ввиду наличия фактора постоянной угрозы общедоступному серверу, независимо от степени важности информационного ресурса, и отсутствия комплексных методик анализа защищенности и организации защиты.

Наиболее распространенным HTTP-сервером является Apache, в большинстве случаев используется платформа PHP в связке с СУБД MySQL.

Когда HTTP-сервер обслуживает единственный сайт, который использует единственную базу данных, безопасность всего сервера зависит от грамотности составления кода используемых скриптов. При экстренной ситуации источник утечки информации достаточно легко выявляется путем анализа журналов HTTP-

сервера и системы, после чего устраняется. Современные сервера обслуживают несколько десятков сайтов, при этом не различаются приоритеты процессов между ними, что в десятки раз повышает риск НСД, при этом, время, затрачиваемое на анализ ситуации и выработку решений, значительно возрастает и напрямую зависит от количества сайтов расположенных на сервере. Незначительная ошибка при написании скрипта приводит к компрометации информации всего сервера.

Основные требования к разграничению доступа, относительно процессов закрепеленных за одним из сайтов:

- невозможность листинга директорий закрепленных за другими сайтами;
- невозможность просмотра файлов закрепленных за другими сайтами.

Очень остро стоит вопрос о совместном использовании ресурсов веб-сервера несколькими пользователями и, в частности, необходимости предоставления услуг хостинга не доверенным лицам, зачастую малоквалифицированным. На данном этапе велика вероятность проникновения в систему злоумышленника с применением методов социальной инженерии, возможны инсайдерские атаки, возможно несанкционированное получение доступа к ресурсам веб-сервера извне через непредумышленно используемые уязвимые сценарии.

Проблема организации и технологии защиты веб-серверов осложняется несовершенством архитектурной модели безопасности существующего программного обеспечения.

Важно защитить целевой сервер от компрометации путем изолирования составляющих компонентов веб-сервера.

Рассмотрим последнюю ветку HTTP сервера Apache и PHP 5 как серверную языковую платформу. Необходимо установить MySQL, затем установить сконфигурировать связку Apache + PHP + MySQL. В качестве модуля превентивной защиты рекомендуется mod_security.

Рекомендации по безопасности, предлагаемые разработчиками Apache, ограничиваются настройкой SSI и CGI механизмов, а также директивами User и Group определяющими от какого пользователя и группы работает веб-сервер. Разработчики PHP делают упор на безопасность внутренней обработки функций. На данный момент используются два основных инструмента для ограничения возможностей скриптов написанных на PHP, это защищенный режим и опция open_basedir.

Защищенный режим в PHP – это попытка решить проблему безопасности на совместно используемых серверах. Представляет собой совокупную настройку ряда параметров и ограничений. Отключает ряд функций, в том числе потенциально необходимых для нормального функционирования сайтов, например fopen().

Второй инструмент – это использование опции open_basedir. Данная опция ограничивает список файлов, которые могут быть открыты в PHP, указанным деревом директорий независимо от того, используется защищенный режим или нет.

Согласно среднестатистическим данным, каждый месяц появляются новые способы обхода приведенных ограничений.

Концептуально неверно решать проблему совместного разграниченного использования ресурсов сервера на уровне PHP. Но поскольку альтернатива на уровне веб-сервера или операционной системы на сегодняшний день отсутствует, проблема остается актуальной.

Частично данную проблему решают за счет различных способов виртуализации. В этом случае для каждого сайта выделяют отдельную операционную систему на одном сервере. Законченная система представляет собой единый сервер, на котором работает несколько независимых систем под управлением одной. Однако

данный способ является высокозатратным с точки зрения аппаратных ресурсов и не рационален при наличии большого числа сайтов.

Концептуально наиболее правильно было бы разграничивать и на уровне процесса, но с дополнительной проверкой на уровне файловой системы. Для данной задачи необходим врапер для PHP процессов и тонкая настройка прав на уровне файловой системы. Рассмотрим пример подобной модели реализуемой под управлением FreeBSD. Теоретически целесообразен вариант запуска PHP процесса с тем же самым пользователем и группой, что и конечный скрипт, соответственно скрипт получит права на самомодификацию. Таким образом, возникает необходимость в пользователе-посреднике.

На сегодняшний день существует врапер, ориентированный на использование с интерпретатором PHP — suPHP, который использует функции set_uid() и set_gid(). По умолчанию данное программное обеспечение устанавливается в режиме «OWNER», то есть процесс для каждого файла будет создан в соответствии с владельцем и группой данного файла. Доступны режимы «FORCE» и «PARANOID». Остановимся подробнее на режиме «FORCE», данный режим разрешает более тонкие настройки конфигурации системы.

Для каждого сайта создаем владельца — owner1, owner2. Создаем соответствующие владельцам группы own_group1, own_group2. Заводим папки сайтов на сервере: site1, site2. Выполняем команду chown -R owner1:own_group1 site1 && chown -R owner2:own_group2 site2 для присвоения директориям сайтов со всем содержимым соответствующих прав. Далее создаем пользователя site1_runner и включаем его в группу own_group1, аналогично создаем пользователя site2_runner и включаем его в группу own_group2. Рекомендуется не создавать домашние каталоги для данных пользователей, использовать nologin для входа и не присваивать пароль. Далее в настройках VirtualHost для site1 используем директиву suPHP_UserGroup site1_runner own_group1 и для site2 соответственно suPHP_UserGroup site2_runner own_group2. Необходимо присвоить всем файлам с расширением PHP права 640. Таким образом, мы полностью изолировали скрипты от чтения средствами скриптов других пользователей. Данная настройка происходит и на уровне процесса, и на уровне файловой системы.

Разработанная пользовательская модель целиком и полностью опирается на права доступа, определенные на уровне файловой системы. Однако используется и разделенное адресное пространство.

Соответствующие ресурсу процессы запускаются от имени специально созданного пользователя-посредника, индивидуального для каждого ресурса.

Используемая маска прав, разрешающая лишь чтение для пользователя группы, не позволяет сценарию возможности модификации доступных файлов, а также запрещает возможность обращения к файлам других пользователей.

Таким образом, применение модели позволило реализовать все требования к разграничению доступа между пользователями.

Подобная надстройка над интерпретатором PHP искажает понимание интерпретатором ряда директив, в частности open_basedir. В результате пользователю запрещен доступ к каталогам и файлам других пользователей, но разрешен свободный листинг оставшейся части файловой системы, за исключением критичных файлов (master.passwd и т.д.).

Проблема разграничения привилегий доступа к пользовательским данным решена, однако остается проблема разрешенного и неконтролируемого доступа к ряду файлов операционной системы, потенциально содержащих чувствительную информацию.

В большинстве случаев не имеется возможности ограничить доступ к данным файлам, поскольку используемые права доступа необходимы для функционирования операционной системы целевого сервера.

Именно данное обстоятельство является основной причиной необходимости вынесения всего веб-сервера в изолированное адресное и физическое пространство, называемое chroot-окружением.

Основной целью применения технологии chroot было максимально обезопасить операционную систему сервера от внешнего воздействия, однако, имеется ряд дополнительных преимуществ:

- Сохранение работоспособности операционной системы сервера.
- Свободный выбор программных средств внутри chroot-окружения и произвольная расстановка прав на их использование.
- Отсутствие файлов содержащих критическую для сервера информацию.
- Возможность быстрого изменения свойств файловой системы при размещении на отдельном разделе (/etc/fstab noexec, nosuid, read only).

Следовательно, возникает необходимость выведения HTTP сервера Apache со всеми модулями в отдельное пространство как адресное, так и физическое. Это позволит максимально обезопасить операционную систему сервера от воздействия извне. Для реализации оптимальным вариантом является mod_chroot. Всё необходимое для нормальной работы HTTP сервера в CHROOT-окружении, а также файлы сайтов выведены в отдельный раздел – /chroot.

Данная система показывает отличные параметры безопасности и сохраняет хорошие скоростные характеристики при значительных нагрузках. СУБД MySQL функционирует в нормальной среде, что позволяет значительно снизить потребление ресурсов. Файлы журналов также находятся вне пределов CHROOT окружения, как и файлы базы данных. Также недоступными для потенциального злоумышленника являются файлы конфигурации Apache. В то же время файл с настройками PHP и файл тонкой подстройки suPHP (не затрагивает модель пользователей). Появилась возможность безболезненно для функционирования сервера удалять системные файлы, например файлы паролей, файлы, отвечающие за исполнение команд (ls, cp, ln и т.д.), потенциально опасные системные файлы с установленным SUID-битом. Также удалось избежать строгих ограничений, накладываемых безопасным режимом. В целом, получаемая конфигурация достаточно гибка.

Данная система практически полностью исключает возможность массового взлома в рамках одного сервера и окончательной потери его функциональности. При активации модуля превентивной защиты mod_security со стандартным набором правил, система показала отличные показатели безопасности при проверке с использованием типовых атак на заведомо уязвимый скрипт, а также успешно прошла ряд тестов на отказ оборудования. Злонамеренные, потенциально опасные запросы были отклонены системой от обработки.

В совокупности с настроенными средствами резервного копирования данная система является пригодной для организации массовых хостинг-систем с обеспечением необходимого уровня безопасности и конфиденциальности данных. Недостатком является определенные затруднения при необходимости обновления компонентов системы.

Настройки системы и каждого отдельно установленного продукта не являются стандартными, что также положительно сказывается на защитных свойствах. Ре-

комендуется заблокировать вывод в заголовках ответа HTTP сервера информацию об используемом программном обеспечении.

Данная система с отработанными и автоматизированными механизмами конфигурации системы, заведения новых доменов и пользователей, единым интерфейсом администрирования, конфигурацией квотирования места на жестком диске, а также системы квотирования системных ресурсов вполне может претендовать на статус коммерческого продукта.

УДК 681.3.053:681.32

Е.П. Тумоян

МЕТОД МОДЕЛИРОВАНИЯ КОМПЬЮТЕРНЫХ АТАК НА ОСНОВЕ ВЕРОЯТНОСТЫХ АВТОМАТОВ*

В последние годы вопросы обеспечения безопасности информационных систем приобретают чрезвычайно большое значение. Одним из наиболее важных направлений данной деятельности является разработка методов и средств, которые позволяют обнаружить факт наличия в программном обеспечении ошибок или недокументированных возможностей, использование которых может привести к нарушению безопасности системы. Поиску уязвимостей на этапе разработки систем уделяется весьма значительное внимание. Существуют методики и программные средства, которые позволяют исключить ошибки использования языка или внешних интерфейсов (Microsoft Prefast, Intel PGO) и устранить значительное количество ошибок связанных с логикой обработки данных, реализацией протоколов и т.д. (PEACH и SPIKE). Однако в настоящее время не существует общепринятых формальных или автоматизированных методов оценки безопасности системы, кроме наиболее простых случаев. Для оценки безопасности систем используются методы penetration testing, которые предполагают экспертную оценку системы со стороны квалифицированных специалистов. Такие исследования обычно занимают значительное время и чрезвычайно дороги. Целью данной работы является разработка модели, которая предоставляет возможности описания атак в условиях динамически меняющейся внешней среды и обеспечивает возможности автоматизированного выполнения атак на целевые системы с целью оценки их безопасности. Решение данной задачи позволит не только снизить время и стоимость оценки, но и повысить точность обнаружения.

Разрабатываемая модель предназначена для исследования атаки как последовательности этапов атаки, которая приводит целевую систему в состояние, необходимо атакующему. Кроме того, исходя из условий, которые являются типичными для компьютерных атак, определим следующие ограничения разрабатываемой модели:

1. Атакующий имеет заданный набор этапов атаки. Этап атаки представляет собой законченное воздействие на атакуемую систему, результаты которого могут быть каким-либо образом оценены атакующим. Воздействие может быть как злонамеренным, например эксплуатацией некоторой уязвимости целевой системы, так и нейтральным, например передачей некоторых допустимых данных.

* Работа выполнена при поддержке гранта РФФИ №07-07-00138а.