Министерство науки и высшего образования Российской Федерации
Алтайский государственный университет

Алябышева Ю. А.
Данько Е. В.
Константинова О. В.
Михеева Т. В.
Половикова О. Н.
Смолякова Л. Л.
Терновой О. С.

Введение в безопасность Web-приложений

Учебное пособие

Барнаул 2020

Об издании – 1, 2

сведения об издании

УДК 004.056.5(075.8)
ББК 32.973.4я73
В 24

Авторы: Алябышева Юлия Анатольевна, Данько Евгений Викторович, Константинова Ольга Владимировна, Михеева Татьяна Викторовна, Половикова Ольга Николаевна, Смолякова Лариса Ленгардовна, Терновой Олег Степанович

Рецензенты: д-р техн. наук, профессор Н. М. Оскорбин

В 24 **Введение в безопасность Web-приложений** : учебное пособие / Ю.А. Алябышева, Е.В. Данько, О.В. Константинова, Т.В. Михеева, О.Н. Половикова, Л.Л. Смолякова, О.С. Терновой ; АлтГУ. – 1 CD-R (2,5 Мб). – Систем. требования: Intel Pentium 1,6 GHz и более ; 512 Мб (RAM) ; Microsoft Windows 7 и выше ; Adobe Reader. – Загл. с титул. экрана. – Текст : электронный.

Учебное электронное издание

В издании рассматриваются вопросы, связанные с безопасной разработкой WEB-приложений. Даются базовые сведения о web-технологиях, освящаются наиболее популярные уязвимости и угрозы, приводятся рекомендации по конфигурированию серверных приложений и безопасной разработке.

© Алябышева Ю.А., Данько Е.В., Константинова О.В., Михеева Т.В., Половикова О.Н., Смолякова Л.Л., Терновой О.С., 2020
© Алтайский государственный университет, 2020

производственно-технические сведения

Публикуется в авторской редакции

Верстка: О. В. Константинова

Дата подписания к использованию: 17.06.2020 г.

Объем издания: 2,5 Мб

Комплектация издания: 1 CD-R

Тираж 15 дисков

ФГБОУ ВО «Алтайский государственный университет»

656049, Барнаул, пр. Ленина, 61

Оглавление

I. Базовые сведения о web-технологиях

I.1. Введение в web-технологии
I.2. О роли стандартизации в интернет
I.3. Стек протоколов TCP/IP
I.4. Структура и принципы WWW
I.5. Обеспечение безопасности передачи данных HTTP

II. Уязвимости по классификации OWASP

III. Основные уязвимости WEB-приложений и способы защиты

III.1. DDoS-атака
III.2. Межсайтовый скриптинг (XSS)
III.3. SQL-инъекции
III.4. Перебор паролей
III.5. Незащищённые ресурсы

IV. Особенности безопасного конфигурирования серверных приложений

IV.1. SSH-доступ
IV.2. FTP-доступ

V. Основы безопасной разработки web-приложений

V.1. Обработка данных форм
V.2. Обработка загрузки файлов
V.3. Аутентификация

VI. Программные средства для обеспечения безопасности web-сервера

VI.1. Web-брандмауэр
VI.2. Использование ModSecurity
VI.3. Контроль целостности

VII. Программные средства для анализа безопасности Web-приложений

VII.1. Сканирование и поиск уязвимостей
VII.2. Проведение нагрузочных тестов

Список используемой литературы

I. Базовые сведения о web-технологиях

I.1. Введение в web-технологии

Что же такое интернет? Всемирная информационная компьютерная сеть, которая связывает между собой как пользователей компьютерных сетей, так и пользователей индивидуальных компьютеров для обмена информацией.

Это самая большая в мире сеть, не имеющая единого центра управления, но работающая по единым правилам и предоставляющая своим пользователям единый набор услуг. Каждой сетью управляет независимый оператор – поставщик услуг интернета (ISP, Internet Service Provider).

С точки зрения пользователей интернет представляет собой набор информационных ресурсов, размещенных по различным сетям, включая ISP-сети, корпоративные сети, сети и отдельные компьютеры домашних пользователей. Каждый отдельный компьютер в данной сети называется хостом (от английского термина host).

Современный интернет обязан своему появлению объединенной сети ARPANET, которая начиналась как скромный эксперимент в новой тогда технологии коммутации пакетов в таблице I.1. Сеть ARPANET была развернута в 1969 г. и состояла поначалу всего из четырех узлов с коммутацией пакетов, используемых для взаимодействия горстки хостов и терминалов. Первые линии связи, соединявшие узлы, работали на скорости всего 50 Кбит/с. Сеть ARPANET финансировалась управлением перспективного планирования научно-исследовательских работ ARPA (Advanced Research Projects Agency) министерства обороны США и предназначалась для изучения технологии и протоколов коммутации пакетов, которые могли бы использоваться для кооперативных распределенных вычислений.

Таблица I.1. Хронология развития интернета (с 1966 по 2022 гг.)

Год	**Событие**
1966	Эксперимент с коммутацией пакетов управления ARPA
1969	Первые работоспособные узлы сети ARPANET
1972	Изобретение распределенной электронной почты
1973	Первые компьютеры, подключенные к сети ARPANET за пределами США
1975	Сеть ARPANET передана в ведение управления связи министерства обороны США
1980	Начинаются эксперименты с TCP/IP
1981	Каждые 20 дней к сети добавляется новый хост
1983	Завершен переход на TCP/IP
1986	Создана магистраль NSFnet
1990	Сеть ARPANET прекратила существование
1991	Появление Gopher
1991	Изобретение Всемирной паутины. Выпущена система PGP. Появление Mosaic
1995	Приватизация магистрали интернета
1996	Построена магистраль OC-3 (155 Мбит/с)
1998	Число зарегистрированных доменных имен превысило 2 млн.
2000	Количество индексируемых web-страниц превысило 1 млрд.
2018	Ежегодный глобальный IP-трафик составлял 1,5 Зб в год, или 122 Эб в месяц
2022	Прогноз: Ежегодный глобальный IP-трафик достигнет 4,8 Зб в год или 396 Эб в месяц.

Компания Cisco Systems, Inc., которая является мировым лидером на рынке сетевых решений для Internet, делает прогноз, что ежегодный глобальный IP-трафик достигнет к 2022 году 4,8 Зетабайт (Зб) в год, или 396 Эксабайт (Эб) в месяц.

В 2017 году годовая скорость для глобального трафика IP составляла 1,5 Зб в год, или 122 Эб в месяц. Глобальный IP-трафик увеличится втрое в течение следующих 5 лет. Ежегодный темп роста (CAGR) составляет 26% с 2017 по 2022 гг. Ежемесячный IP-трафик достигнет 50 Гб на душу населения к 2022 г., по сравнению с 16 Гб на душу населения в 2017 г. Количество устройств, подключенных к IP-сетям, будет более чем в 3 раза больше в 2022 г. К 2022 г. на душу населения будет приходиться 3,6 сетевых устройства, по сравнению с 2,4 сетевыми устройствами на душу населения в 2017 г. К 2022 г. сетевые устройства будут насчитывать 28,5 млрд. против 18 млрд. в 2017 г. Трафик смартфонов будет превышать траффик персональных компьютеров. В 2018 г. на долю персональных компьютеров приходилось 41% от общего IP-трафика, а в 2022 г. – персональных компьютеров будет приходиться только 19% IP-трафика. На смартфоны будет приходиться 44% общего IP-трафика к 2022 г. по сравнению с 18% в 2018 г.

Возможно ли централизованное управление в такой глобальной сети? Ответ на данный вопрос будет отрицательным, поскольку, во-первых, данная сеть является транснациональной и, во-вторых, в силу исторических предпосылок ее формирования.

Тем не менее, в интернете могут проявляться опосредованные формы централизации в форме единой технической политики, согласованном наборе технических стандартов, назначении имен и адресов компьютеров и сетей, входящих в интернет.

То есть интернет является децентрализованной сетью, что имеет свои достоинства и недостатки.

1. Достоинства:
    - Легкость наращивания интернета путем заключения соглашения между двумя ISP.
2. Недостатки:
    - Сложность модернизации технологий и услуг интернета, поскольку требуются согласованные усилия всех поставщиков услуг.
    - Невысокая надежность услуг интернета.
    - Ответственность за работоспособность отдельных сегментов этой сети возлагается на поставщиков услуг интернета.

Существуют различные типы поставщиков услуг интернета:

- Поставщик услуг интернета выполняет транспортную функцию для конечных пользователей – передачу их трафика в сети других поставщиков услуг интернета.
- Поставщик интернет-контента имеет собственные информационно-справочные ресурсы, предоставляя их содержание в виде web-сайтов.
- Поставщик услуг хостинга предоставляет свои помещения, каналы связи и серверы для размещения внешнего контента.
- Поставщик услуг по доставке контента занимается только доставкой контента в многочисленные точки доступа с целью повышения скорости доступа пользователей к информации.
- Поставщик услуг по поддержке приложений предоставляет клиентам доступ к крупным универсальным программным продуктам, например SAP R3.
- Поставщик биллинговых услуг обеспечивает оплату счетов по интернету.

I.2. О роли стандартизации в интернет

Как следует из всего вышеизложенного, интернет является очень сложной сетью, и соответственно такой же сложной является задача организации взаимодействия между устройствами сети. Для решения такого рода задач используется декомпозиция, т.е. разбиение сложной задачи на несколько более простых задач-модулей. Одной из концепций, реализующих декомпозицию, является многоуровневый подход. Такой подход дает возможность проводить разработку, тестирование и модификацию каждого отдельного уровня независимо от других уровней. Иерархическая декомпозиция позволяет, перемещаясь в направлении от более низких к более высоким уровням переходить к более простому представлению решаемой задачи.

Специфика многоуровневого представления сетевого взаимодействия состоит в том, что в процессе обмена сообщениями участвуют как минимум две стороны, для которых необходимо обеспечить согласованную работу двух иерархий аппаратно-программных средств. Каждый из уровней должен поддерживать интерфейс с выше и нижележащими уровнями собственной иерархии средств и интерфейс со средствами взаимодействия другой стороны на том же уровне иерархии. Данный тип интерфейса называется протоколом (рис. I.1.).

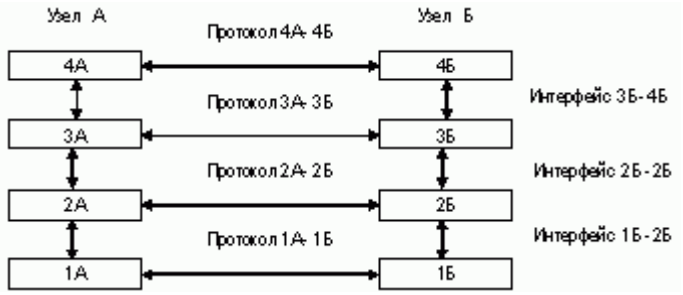


Рис. I.1. Организация взаимодействия между уровнями иерархии при иерархической декомпозиции в сети интернет

Иерархически организованный набор протоколов, достаточный для организации взаимодействия узлов в сети, называется стеком протоколов.

В начале 80-х гг. международные организации по стандартизации ISO (International Organization for Standardization), ITU (International

Telecommunications Union) и другие разработали стандартную модель взаимодействия открытых систем OSI (Open System Interconnection). Назначение данной модели состоит в обобщенном представлении средств сетевого взаимодействия. Ее также можно рассматривать в качестве универсального языка сетевых специалистов (справочной модели).

Поскольку сеть – это соединение разнородного оборудования, актуальной является проблема совместимости, что в свою очередь, требует согласования всеми производителями общепринятых стандартов. Открытой является система, построенная в соответствии с открытыми спецификациями.

Спецификация представляет собой формализованное описание аппаратных (программных) компонентов, способов их функционирования, взаимодействия с другими компонентами, условий эксплуатации, особых характеристик. Под открытыми спецификациями понимаются опубликованные, общедоступные спецификации, соответствующие стандартам и принятые в результате достижения согласия после всестороннего обсуждения всеми заинтересованными сторонами. Использование открытых спецификаций при разработке систем позволяет третьим сторонам разрабатывать для этих систем аппаратно-программные средства расширения и модификации, а также создавать программно-аппаратные комплексы из продуктов разных производителей.

Если две сети построены с соблюдением принципов открытости, это дает следующие преимущества:

- Возможность построения сети из аппаратных и программных средств различных производителей, придерживающихся стандарта.
- Безболезненная замена отдельных компонентов сети другими, более совершенными.
- Легкость сопряжения одной сети с другой.

В рамках модели OSI средства взаимодействия делятся на семь уровней: прикладной, представления, сеансовый, транспортный, сетевой, канальный и физический. В распоряжение программистов предоставляется прикладной про-

граммный интерфейс, позволяющий обращаться с запросами к самому верхнему уровню, а именно, уровню приложений.

Сеть интернет строилась в полном соответствии с принципами открытых систем. В разработке стандартов этой сети принимали участие тысячи специалистов-пользователей сети из вузов, научных организаций и компаний. Результат работы по стандартизации воплощается в документах RFC.

Таблица I.2. Примеры популярных RFC-документов

Номер RFC	**Тема**
RFC 768	UDP
RFC 791	IP
RFC 793	TCP
RFC 822	Формат электронной почты, заменен RFC 2822
RFC 959	FTP
RFC 1034	DNS – концепция
RFC 1035	DNS – внедрение
RFC 1591	Структура доменных имен
RFC 1738	URL
RFC 1939	Протокол POP версии 3 (POP3)
RFC 2026	Процесс стандартизации в интернете
RFC 2045	MIME
RFC 2231	Кодировка символов
RFC 2616	HTTP
RFC 2822	Формат электронной почты
RFC 3501	IMAP версии 4 издание 1 (IMAP4rev1)

RFC (англ. Request for Comments) – документ из серии пронумерованных информационных документов интернета, содержащих технические спецификации и Стандарты, широко применяемые во Всемирной сети. В настоящее время первичной публикацией документов RFC занимается IETF под эгидой открытой организации общество интернета (ISOC). Правами на RFC обладает именно общество интернет. Формат RFC появился в 1969 г. При обсуждении проекта ARPANET. Первые RFC распространялись в печатном виде на бумаге в виде обычных писем, но уже с декабря 1969 г., когда заработали первые сегменты ARPANET, документы начали распространяться в электронном виде. В таблице I.2. приведены некоторые из наиболее известных документов RFC.

Основным организационным подразделением, координирующим работу по стандартизации интернет, является ISOC (Internet Society), объединяющее порядка 100 тысяч участников, которые занимаются различными аспектами развития данной сети. ISOC курирует работу IAB (Internet Architecture Board), включающую две группы:

- IRTF (Internet Research Task Force). Координирует долгосрочные исследовательские проекты, относящиеся к TCP/IP.
- IETF (Internet Engineering Task Force). Инженерная группа, определяющая спецификации для последующих стандартов интернет.

Разработкой стандартов для сети web, начиная с 1994 года, занимается Консорциум W3C (World Wide Web Consortium), основанный и до сих пор возглавляемый Тимом Бернерсом-Ли. Консорциум W3C — организация, разрабатывающая и внедряющая технологические стандарты для интернета и WWW. Миссия W3C формулируется следующим образом: «Полностью раскрыть потенциал Всемирной паутины путём создания протоколов и принципов, гарантирующих долгосрочное развитие Сети». Две другие важнейшие задачи Консорциума – обеспечить полную «интернационализацию Сети» и сделать её доступной для людей с ограниченными возможностями.

В российском сегменте интернета имеется своя организация – Российский НИИ Развития Общественных Сетей РОСНИИРОС (Russian Institute for Public

Networks, RIPN). РОСНИИРОС занимается координацией российских исследований и разработок в интернете.

Прежде чем перейти к описанию структуры, принципов работы и основных протоколов сети web, рассмотрим основной стек протоколов сети интернет – стек TCP/IP.

I.3. Стек протоколов TCP/IP

Эти протоколы изначально ориентированы на глобальные сети, в которых качество соединительных каналов неидеально. Он позволяет создавать глобальные сети, компьютеры в которых соединены друг с другом самыми разными способами от высокоскоростных оптоволоконных кабелей и спутниковых каналов до коммутируемых телефонных линий. TCP/IP соответствует модели OSI достаточно условно и содержит 4 уровня. Прикладной уровень стека соответствует трём верхним уровням модели OSI: прикладному, уровню представления и сеансовому.

В сети данные всегда передаются блоками относительно небольшого размера. Каждый блок имеет префиксную часть (заголовок), описывающую содержимое блока, и суффиксную, содержащую, например, информацию для контроля целостности передаваемого блока данных.

Название стека протоколов TCP/IP состоит из названий двух разных протоколов. Протокол IP (Internet Protocol) представляет собой протокол нижнего (сетевого) уровня и отвечает за передачу пакетов данных в сети. Он относится к протоколам датаграмм и работает без подтверждений. Последнее означает, что при его использовании доставка пакетов данных не гарантируется и не подтверждается. Не гарантируется также и то, что пакеты достигнут пункта назначения в той последовательности, в которой они были отправлены.

К протоколам сетевого уровня относится также протокол межсетевых управляющих сообщений ICMP (Internet Control Message Protocol), предназначенный для передачи маршрутизатором источнику информации об ошибках при передаче пакета.

Очевидно, что намного удобнее передавать данные по каналу, который работает корректно, доставляя все пакеты по порядку. Поэтому над протоколом IP работает протокол передачи данных более высокого (транспортного) уровня – TCP (Transmission Control Protocol). Посылая и принимая пакеты через протокол IP, протокол TCP гарантирует доставку всех переданных пакетов данных в правильной последовательности.

Следует отметить, что при использовании протокола IP обеспечивается более быстрая передача данных, так как не тратится время на подтверждение приёма каждого пакета. Есть и другие преимущества. Одно из них заключается в том, что он позволяет рассылать пакеты данных в широковещательном режиме, при котором они достигают всех компьютеров физической сети. Что же касается протокола TCP, то для передачи данных с его помощью необходимо создать канал связи между компьютерами. Он и создаётся с использованием протокола IP.

Для идентификации сетевых интерфейсов используются 3 типа адресов:

- аппаратные адреса (или MAC-адреса);
- сетевые адреса (IP-адреса);
- символьные (доменные) имена.

В рамках IP протокола для создания глобальной системы адресации, не зависящей от способов адресации узлов в отдельных сетях, используется пара идентификаторов, состоящая из номера сети и номера узла. При этом IP-адрес идентифицирует не отдельный компьютер или маршрутизатор, а одно сетевое соединение в составе сети, в которую он входит; то есть конечный узел может входить в несколько IP-сетей.

Система доменных имён DNS

Аппаратное и программное обеспечение в рамках TCP/IP сетей для идентификации узлов использует IP-адреса, пользователи предпочитают символьные имена (доменные имена).

Для установления соответствия между доменным именем и IP-адресом используется специальная система доменных имен (DNS, Domain Name System), которую создают администраторы сети с помощью таблиц соответствия.

В сетях TCP/IP используется доменная система имён, имеющая иерархическую структуру. Запись доменного имени начинается с самой младшей составляющей, затем после точки следует следующая по старшинству символьная часть имени и так далее. Последовательность заканчивается корневым именем, например: company.mail.ru.

Построенная таким образом система имён позволяет разделять административную ответственность по поддержке уникальности имён в пределах своего уровня иерархии между различными людьми или организациями.

Совокупность имён, у которых несколько старших составных частей совпадают, образуют домен имён.

Корневой домен управляется центральными органами интернета: IANA и InterNIC.

Домены верхнего уровня назначаются для каждой страны, а также для различных типов организаций. Имена этих доменов должны следовать международному стандарту ISO 3166. Для обозначения стран используются двухбуквенные аббревиатуры, например, ru – Российская Федерация, us – США, it – Италия, fr – Франция.

Для различных типов организаций используются трёхбуквенные аббревиатуры:

- net – сетевые организации;
- org – некоммерческие организации;
- com – коммерческие организации;
- edu – образовательные организации;
- gov – правительственные организации.

Администрирование каждого домена возлагается на отдельную организацию, которая делегирует администрирование поддоменов другим организациям.

Для получения доменного имени необходимо зарегистрироваться в соответствующей организации, которой организация InterNIC делегировала свои полномочия по распределению доменных имён.

Регистратором доменных имён в зоне ru до 2005 г. Являлся Российский научно-исследовательский институт развития общественных сетей (РосНИИРОС). В настоящее время регистрация доменов осуществляется одним из действующих регистраторов.

В TCP/IP сетях соответствие между доменными именами и IP-адресами может устанавливаться как локальными средствами, так и централизованными службами. Первоначально соответствие задавалось с помощью создаваемого вручную на хосте файла hosts.txt, состоящего из строк, содержащих пару вида «доменное имя – IP-адрес». Однако с активным ростом интернета такое решение оказалось не масштабируемым.

Альтернативное решение – централизованная служба DNS, использующая распределённую базу отображений «доменное имя – IP-адрес». Сервер домена хранит только имена, которые заканчиваются на следующем ниже по дереву уровне. Это позволяет распределять более равномерно нагрузку по разрешению имён между всеми DNS-серверами. Каждый DNS-сервер помимо таблицы отображения имён содержит ссылки на DNS-серверы своих поддоменов.

Существуют две схемы разрешения DNS-имен.

Нерекурсивная процедура:

1. DNS-клиент обращается к корневому DNS-серверу с указанием полного доменного имени.
2. DNS-сервер отвечает клиенту, указывая адрес следующего DNS-сервера, обслуживающего домен верхнего уровня, заданный в следующей старшей части имени.

3. DNS-клиент делает запрос следующего DNS-сервера, который отсылает его к DNS-серверу нужного поддомена и т.д., пока не будет найден DNS-сервер, в котором хранится соответствие запрошенного имени IP-адресу. Сервер даёт окончательный ответ клиенту.

Рекурсивная процедура:

DNS-клиент запрашивает локальный DNS-сервер, обслуживающий поддомен, которому принадлежит клиент, далее, если локальный DNS-сервер знает ответ, он возвращает его клиенту. Если локальный сервер не знает ответ, то он выполняет итеративные запросы к корневому серверу. После получения ответа сервер передаёт его клиенту.

Таким образом, при рекурсивной процедуре клиент фактически перепоручает работу своему серверу. Для ускорения поиска IP-адресов DNS-серверы широко применяют кэширование (на время от часов до нескольких дней) проходящих через них ответов.

I.4. Структура и принципы WWW

Сеть WWW образуют миллионы web-серверов, расположенных по всему миру. Web-сервер является программой, запускаемой на подключённом к сети компьютере и передающей данные по протоколу HTTP.

Для идентификации ресурсов (зачастую файлов или их частей) в WWW используются идентификаторы ресурсов URI (Uniform ResourceIdentifier). Для определения местонахождения ресурсов в этой сети используются локаторы ресурсов URL (Uniform Resource Locator). Такие URL-локаторы представляют собой комбинацию URI и системы DNS.

Доменное имя (или IP-адрес) входит в состав URL для обозначения компьютера (его сетевого интерфейса), на котором работает программа web-сервер.

На клиентском компьютере для просмотра информации, полученной от web-сервера, применяется специальная программа – web-браузер. Основная функция web-браузера – отображение гипертекстовых страниц (web-страниц).

Для создания гипертекстовых страниц в WWW изначально использовался язык HTML. Множество web-страниц образуют web-сайт.

Прокси-серверы

Прокси-сервер (proxy-server) – служба в компьютерных сетях, позволяющая клиентам выполнять косвенные запросы к другим сетевым службам.

Сначала клиент подключается к прокси-серверу и запрашивает какой-либо ресурс, расположенный на другом сервере. Затем прокси-сервер либо подключается к указанному серверу и получает ресурс у него, либо возвращает ресурс из собственного кэша (если имеется). В некоторых случаях запрос клиента или ответ сервера может быть изменён прокси-сервером в определённых целях. Также прокси-сервер позволяет защищать клиентский компьютер от некоторых сетевых атак.

Чаще всего прокси-серверы применяются для следующих целей:

- обеспечение доступа с компьютеров локальной сети в интернет;
- кеширование данных: если часто происходят обращения к одним и тем же внешним ресурсам, то можно держать их копию на прокси-сервере и выдавать по запросу, снижая тем самым нагрузку на канал во внешнюю сеть и ускоряя получение клиентом запрошенной информации;
- сжатие данных: прокси-сервер загружает информацию из интернета и передаёт информацию конечному пользователю в сжатом виде;
- защита локальной сети от внешнего доступа: например, можно настроить прокси-сервер так, что локальные компьютеры будут обращаться к внешним ресурсам только через него, а внешние компьютеры не смогут обращаться к локальным вообще (они «видят» только прокси-сервер);
- ограничение доступа из локальной сети к внешней: например, можно запретить доступ к определённым web-сайтам, ограничить использование интернета каким-то локальным пользователям, уста-

навливать квоты на трафик или полосу пропускания, фильтровать рекламу и вирусы;

- анонимизация доступа к различным ресурсам. Прокси-сервер может скрывать сведения об источнике запроса или пользователе. В таком случае целевой сервер видит лишь информацию о прокси-сервере, например, IP-адрес, но не имеет возможности определить истинный источник запроса. Существуют также искажающие прокси-серверы, которые передают целевому серверу ложную информацию об истинном пользователе.

Протоколы интернет прикладного уровня

Самый верхний уровень в иерархии протоколов интернет занимают следующие протоколы прикладного уровня:

- DNS – распределённая система доменных имён, которая по запросу, содержащему доменное имя хоста, сообщает IP адрес;
- HTTP – протокол передачи гипертекста в интернет;
- HTTPS – расширение протокола HTTP, поддерживающее шифрование;
- FTP (File Transfer Protocol – RFC 959) – протокол, предназначенный для передачи файлов в компьютерных сетях;
- Telnet (TELecommunication NETwork – RFC 854) – сетевой протокол для реализации текстового интерфейса по сети;
- SSH (Secure Shell – RFC 4251) – протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и передачу файлов. В отличие от Telnet шифрует весь трафик;
- POP3 – протокол почтового клиента, который используется почтовым клиентом для получения сообщений электронной почты с сервера;
- IMAP – протокол доступа к электронной почте в интернет;

- SMTP – протокол, который используется для отправки почты от пользователей к серверам и между серверами для дальнейшей пересылки к получателю;
- LDAP – протокол для доступа к службе каталогов X.500, является широко используемым стандартом доступа к службам каталогов;
- XMPP (Jabber) – основанный на XML расширяемый протокол для мгновенного обмена сообщениями в почти реальном времени;
- SNMP – базовый протокол управления сети Internet.

Рассмотрим более подробно некоторые из этих протоколов.

FTP

FTP позволяет подключаться к серверам FTP, просматривать содержимое каталогов и загружать файлы с сервера или на сервер; кроме того, возможен режим передачи файлов между серверами; FTP позволяет обмениваться файлами и выполнять операции над ними через TCP-сети. Данный протокол работает независимо от операционных систем. Исторически протокол FTP предложил открытую функциональность, обеспечивая прозрачный перенос файлов с одного компьютера на другой по сети. Это не так тривиально, как может показаться, так как у разнотипных компьютеров могут различаться размеры слов, биты в словах могут храниться в неодинаковом порядке или использоваться разные форматы слов.

Telnet

Название «telnet» имеют также некоторые утилиты, реализующие клиентскую часть протокола. Протокол telnet работает в соответствии с принципами архитектуры «клиент-сервер» и обеспечивает эмуляцию алфавитно-цифрового терминала, ограничивая пользователя режимом командной строки. Приложение telnet предоставило язык для общения терминалов с удалёнными компьютерами. Когда появилась сеть ARPANET, для каждой компьютерной системы требовались собственные терминалы. Приложение telnet стало общим знаменателем для терминалов. Достаточно было написать для каждого компьютера

программное обеспечение, поддерживающее «терминал telnet», чтобы один терминал мог взаимодействовать с компьютерами всех типов.

SSH

Сходен по функциональности с протоколами telnet и rlogin, но, в отличие от них, шифрует весь трафик, включая и передаваемые пароли. SSH-клиенты и SSH-серверы имеются для большинства операционных систем.

Почтовые протоколы

Хотя telnet и FTP были и остаются полезными, первым приложением, совершившим переворот в сознании пользователей компьютеров сети ARPANET, стала электронная почта. До сети ARPANET существовали системы электронной почты, но все они были одно компьютерными системами. В 1972 г. Рэй Томлинсон (Ray Tomlinson) из компании BBN написал первый пакет, предоставляющий распределённые почтовые услуги в компьютерной сети из нескольких компьютеров. Уже к 1973 г. Исследования управления ARPA показали, что три четверти всего трафика сети ARPANET составляла электронная почта. Польза электронной почты оказалась столь велика, что все больше пользователей стремилось подключиться к сети ARPANET, в результате чего возрастала потребность в добавлении новых узлов и использовании высокоскоростных линий. Почта популярна и в настоящее время.

- POP3 (Post Office Protocol Version 3 – RFC 1939) – протокол, который используется почтовым клиентом для получения сообщений электронной почты с почтового сервера;
- IMAP (Internet Message Access Protocol – RFC 3501) – протокол доступа к электронной почте. Аналогичен POP3, однако предоставляет пользователю богатые возможности для работы с почтовыми ящиками, находящимися на центральном сервере. Электронными письмами можно манипулировать с компьютера пользователя (клиента) без необходимости постоянной пересылки с сервера и обратно файлов с полным содержанием писем;

- SMTP (Simple Mail Transfer Protocol – RFC 2821) – протокол, предназначенный для передачи электронной почты. Используется для отправки почты от пользователей к серверам и между серверами для дальнейшей пересылки к получателю. Для приема почты почтовый клиент должен использовать протоколы POP3 или IMAP.

Базовым протоколом сети гипертекстовых ресурсов web является протокол HTTP. В его основу положено взаимодействие «клиент-сервер», то есть предполагается, что:

1. Потребитель-клиент, инициировав соединение с поставщиком-сервером, посылает ему запрос.
2. Поставщик-сервер, получив запрос, производит необходимые действия и возвращает обратно клиенту ответ с результатом.

При этом возможны два способа организации работы компьютера-клиента:

- Тонкий клиент – это компьютер-клиент, который переносит все задачи по обработке информации на сервер. Примером тонкого клиента может служить компьютер с браузером, использующийся для работы с web-приложениями.
- Толстый клиент, напротив, производит обработку информации независимо от сервера, использует последний в основном лишь для хранения данных.

Прежде чем перейти к конкретным клиент-серверным web-технологиям, рассмотрим основные принципы и структуру базового протокола HTTP.

HTTP (HyperText Transfer Protocol – RFC 1945, RFC 2616) – протокол прикладного уровня для передачи гипертекста.

Центральным объектом в HTTP является ресурс, на который указывает URL в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы. Особенностью протокола HTTP является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т.д. Именно благодаря возможности

указания способа кодирования сообщения клиент и сервер могут обмениваться двоичными данными, хотя изначально данный протокол предназначен для передачи символьной информации. На первый взгляд это может показаться излишней тратой ресурсов. Действительно, данные в символьном виде занимают больше памяти, сообщения создают дополнительную нагрузку на каналы связи, однако подобный формат имеет много преимуществ. Сообщения, передаваемые по сети, удобочитаемы, и, проанализировав полученные данные, системный администратор может легко найти ошибку и устранить её. При необходимости роль одного из взаимодействующих приложений может выполнять человек, вручную вводя сообщения в требуемом формате.

В отличие от многих других протоколов, HTTP является протоколом без памяти. Это означает, что протокол не хранит информацию о предыдущих запросах клиентов и ответах сервера. Компоненты, использующие HTTP, могут самостоятельно осуществлять сохранение информации о состоянии, связанной с последними запросами и ответами. Например, клиентское web-приложение, посылающее запросы, может отслеживать задержки ответов, а web-сервер может хранить IP-адреса и заголовки запросов последних клиентов.

Все программное обеспечение для работы с протоколом HTTP разделяется на три основные категории:

1. Серверы – поставщики услуг хранения и обработки информации (обработка запросов).
2. Клиенты – конечные потребители услуг сервера (отправка запросов).
3. Прокси-серверы для поддержки работы транспортных служб.

Основными клиентами являются браузеры, например: Chrom, Opera, Mozilla Firefox и другие. Наиболее популярными реализациями web-серверов являются: Internet Information Services (IIS), Apache, Lighttpd, Nginx. Наиболее известные реализации прокси-серверов: Squid, UserGate, Multiproxy, Naviscope.

Протокол HTTP находится на седьмом, прикладном уровне модели OSI в таблице I.3. и работает на основе протокола TCP.

Поскольку протокол HTTP находится на прикладном уровне, прикладные приложения могут использовать непосредственно его для организации сетевого взаимодействия. Кроме того, протокол HTTP является важнейшей частью web-приложений. В этом случае браузер, используя возможности HTTP, взаимодействует с сервером для получения необходимых данных.

Таблица I.3. Модель OSI

Прикладной	HTTP, FTP, POP3, DNS
Представления	
Сеансовый	
Транспортный	TCP
Сетевой	IP
Канальный	Ethernet
Физический	Ethernet

Протокол HTTP предполагает передачу данных в режиме «запрос-ответ». При этом в рамках такого взаимодействия могут передаваться данные практически любого типа – обычный текст, гипертекст (HTML), таблицы стилей, клиентские сценарии, изображения, документы в различных форматах, бинарная информация и т.д.

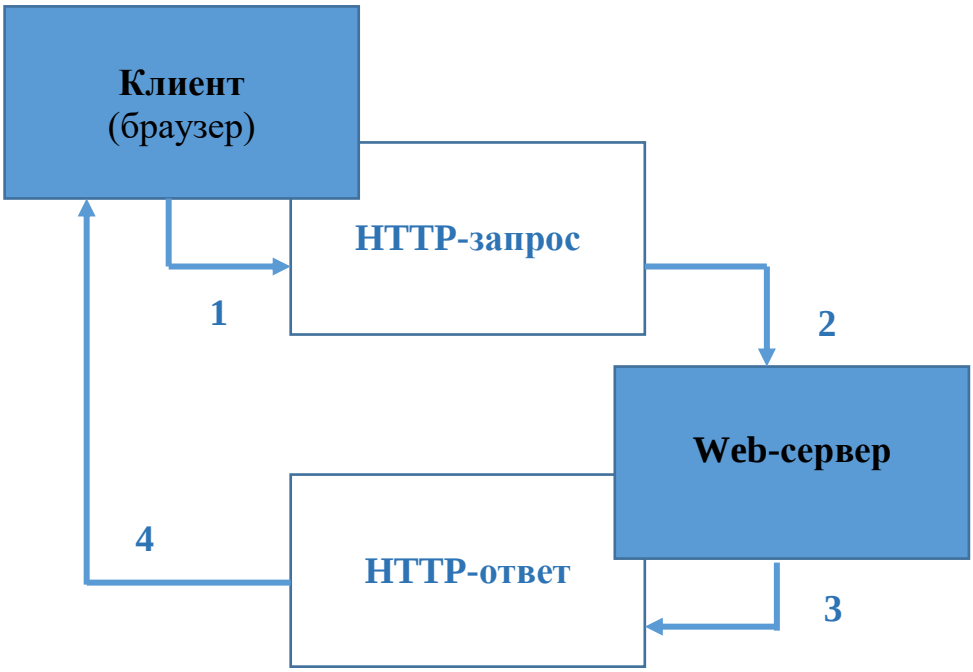


Рис. I.2. HTTP-взаимодействие

В рамках протокола HTTP всегда чётко выделяется клиент и сервер. Клиент всегда является инициатором взаимодействия. Сервер, в свою очередь, прослушивает все входящие соединения и обрабатывает каждое из них. Поскольку HTTP-взаимодействие функционирует по схеме «запрос-ответ», то для инициации сеанса передачи данных необходимо сгенерировать HTTP-запрос.

В рамках этого запроса клиент описывает то, какой ресурс он хочет получить от сервера, а также указывает различные дополнительные параметры. После этого запрос отправляется серверу и тот, в свою очередь, обрабатывает запрос и генерирует HTTP-ответ, в котором содержится служебная информация и содержимое того ресурса, который был запрошен. В целом схематически процесс можно изобразить следующим образом.

HTTP-запрос и HTTP-ответ похожи по своей структуре и называются HTTP-сообщениями. Фактически, все взаимодействие в рамках протокола HTTP сводится к пересылке HTTP-сообщений. Каждое HTTP-сообщение является обычной текстовой информацией, представленной в определённом формате.

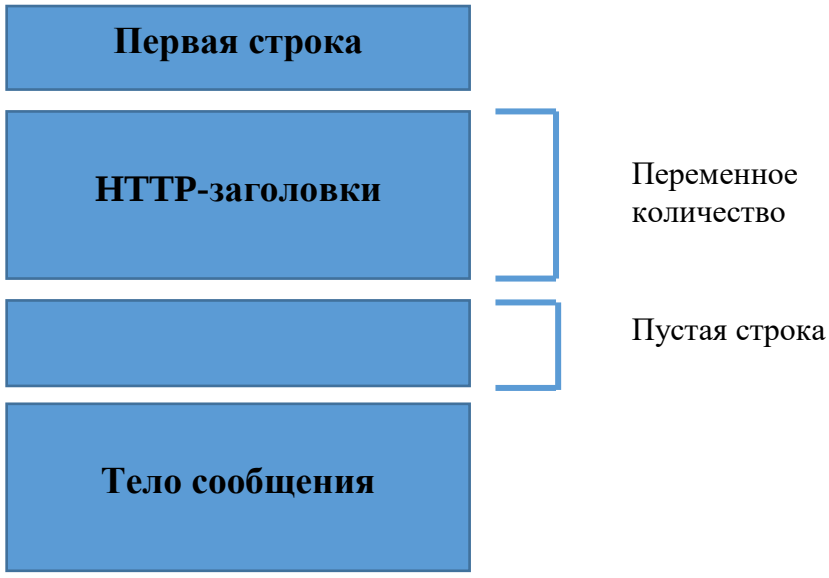


Рис. I.3. HTTP-сообщение

Каждое HTTP-сообщение состоит из нескольких строк. Первой строкой всегда идёт приветственная строка, она существенно различается для HTTP-запроса и HTTP-ответа. Обычно в ней содержится общая информация о запро-

се. После первой строки в HTTP-сообщении присутствуют HTTP-заголовки – каждый заголовок с новой строки. HTTP-заголовки присутствуют как в HTTP-запросе, так и в HTTP-ответе. Смысл HTTP-заголовков заключается в уточнении HTTP-сообщения для того, чтобы принимающая это HTTP-сообщение сторона могла наиболее точно обработать входящее сообщение. Количество заголовков HTTP-сообщения является переменным и зависит от конкретного HTTP-сообщения. Если отправляющая сторона считает, что этот HTTP-заголовок необходим в этом HTTP-сообщении, то она добавляет его, если нет – то не добавляет. Каждый HTTP-заголовок начинается с новой строки. HTTP-заголовок состоит из имени и значения, имя заголовка определяет его предназначение. После набора HTTP-заголовков следует пустая строка, после которой идёт тело HTTP-сообщения (рис.I.3.).

HTTP-запрос формируется на клиенте и отправляется на сервер с целью получения информации от него. В нем содержится информация о ресурсе, который необходимо загрузить, а также дополнительные сведения. Первая строка содержит метод запроса, имя ресурса (с указанием относительного пути на сервере), а также версию протокола. Например, вид приветственной строки может быть определён как «GET /images/men.jpg HTTP/1.1». Такой запрос обращается к серверу с требованием выдать методом GET изображение, расположенное в папке «images» и имеющее название «men.jpg». В HTTP-заголовках указывается уточняющая информация о запросе – версия браузера, возможности клиента принимать сжатое содержимое, возможности кэширования и другие важные параметры, которые могут влиять на формирование ответа. В теле HTTP-запроса обычно содержится информация, которую необходимо передать на сервер. Например, если требуется загрузить файл на сервер, то содержимое файла будет находиться в теле HTTP-запроса. Следует отметить, что размещение данных в теле HTTP-запроса допускается не для всех HTTP-методов. Например, тело HTTP-запроса всегда пустое, если используется метод GET. Таким образом, стандартный HTTP-запрос может выглядеть следующим образом:

GET/images/men.jpg/1.1
Host: icrosoft.com
Encoding: gzip, deflate
Accept-Language: en-us
UserAgent: Google Chrome (compatible; MSIE 5.0; Windows 7)

В приведённом HTTP-запросе клиент обращается к серверу «microsoft.com», запрашивает ресурс «images/men.jpg» и указывает, что он способен принимать сжатое содержимое по алгоритму «gzip» или «deflate», его языком является английский язык и указывает версию своего браузера. Как было отмечено ранее, количество и набор заголовков может существенно отличаться.

POST/images/document.txt HTTP/1.1
Host: icrosoft.com
Encoding: gzip, deflate
Accept-Language: en-us
UserAgent: Google Chrome (compatible; MSIE 5.0; Windows 7)

==xzwaefhfjgjhh67vbcvb777cvbvxc7vcbxcvcbvb787xcbvb76cxvbcvz7cvzcv7c6778zc6v78v7bz8cv67vc678b6c7vc7z766z7cv6b7zc8vb7c

Этот запрос отличается от предыдущего тем, что в нем используется метод POST, который также загружает данные на сервер. При этом сами данные содержаться в теле HTTP-запроса после пустой строки.

HTTP-ответ генерируется web-сервером на поступивший HTTP-запрос. По своей структуре он схож с HTTP-запросом, но имеет определённые отличия. Главное отличие содержится в первой строке. Вместо имени запрашиваемого ресурса и метода запроса в ней указывается статус ответа. Статус указывает на то, насколько успешно выполнился HTTP-запрос. Так в случае, если документ найден на сервере и может быть выдан клиенту, то статус имеет значение «OK», запрос выполнился успешно. Существуют и другие ситуации: документ отсутствует на сервере или у пользователя отсутствуют права на получение ресурса. Таким образом, первая строка HTTP-ответа может принимать значение «HTTP/1.1 200 OK». HTTP-заголовки в HTTP-ответе также являются важным элементом. Они характеризуют содержимое, которое передаётся клиенту.

Например, в этих HTTP-заголовках может содержаться информация о типе содержимого (HTML-документ, изображение и т.д.), длине содержимого (размер в байтах), дате модификации, режиме кэширования и др. Все эти заголовки влияют на способ отображения данных на клиенте, а также устанавливают правила хранения данных в клиентском кэше. Типичный вид HTTP-ответа может быть следующим:

```
HTTP/1.1 200 Ok
Content-Type: text/html
Content-Length: 20101
Last-Modified: Thu, 4 Apr 2019 13:20:10 GMT

<html><head><title>page</title></head>
<body>…</body></html>
```

В приведённом примере сервер указывает, что ресурс найден, его тип – HTML-документ, а также указывает размер и дату модификации. После пустой строки идёт содержимое HTML-документа, т.е. то, что запрашивал клиент. Как и в случае с HTTP-запросом, в HTTP-ответе количество заголовков может изменяться на усмотрение web-сервера.

Поля заголовка, следующие за строкой состояния, позволяют уточнять запрос, т.е. передавать серверу дополнительную информацию. Поле заголовка имеет следующий формат:

Назначение поля определяется его именем, которое отделяется от значения двоеточием.

Имена некоторых наиболее часто встречающихся в запросе клиента полей заголовка, и их назначение приведены в таблице I.4.

При рассмотрении структуры HTTP-запроса было затронуто понятие метода HTTP-запроса. Метод HTTP-запроса определяет, каким образом будет обрабатываться указанный HTTP-запрос. Поскольку HTTP-запросы могут иметь самый разнообразный смысл, то указание метода является важной частью построения HTTP-запроса. HTTP-запросы могут иметь следующие значения: запрос ресурса от сервера, создание или изменение ресурса на сервере, удаление ресурса на сервере и т.д.

Таблица I.4. Поля заголовка запроса HTTP

Поля заголовка HTTP -запроса	**Значение**
Host	Доменное имя или IP-адрес узла, к которому обращается клиент
Referer	URL документа, который ссылается на ресурс, указанный в строке состояния
From	Адрес электронной почты пользователя, работающего с клиентом
Accept	MIME-типы данных, обрабатываемых клиентом. Это поле может иметь несколько значений, отделяемых одно от другого запятыми. Часто поле заголовка Accept используется для того, чтобы сообщить серверу о том, какие типы графических файлов поддерживает клиент
Accept-Language	Набор двух символьных идентификаторов, разделенных запятыми, которые обозначают языки, поддерживаемые клиентом
Accept-Charset	Перечень поддерживаемых наборов символов
Content-Type	MIME-тип данных, содержащихся в теле запроса (если запрос не состоит из одного заголовка)
Content-Length	Число символов, содержащихся в теле запроса (если запрос не состоит из одного заголовка)
Range	Присутствует в том случае, если клиент запрашивает не весь документ, а лишь его часть
Connection	Используется для управления TCP-соединением. Если в поле содержится Close, это означает, что после обработки запроса сервер должен закрыть соединение. Значение Keep-Alive предлагает не закрывать TCP-соединение, чтобы оно могло быть использовано для последующих запросов
User-Agent	Информация о клиенте

Наиболее распространёнными методами HTTP-запроса являются следующие типы методов:

- GET – позволяет получить информацию от сервера, тело запроса всегда остаётся пустым;
- HEAD – аналогичен GET, но тело ответа остаётся всегда пустым, позволяет проверить доступность запрашиваемого ресурса и прочитать HTTP-заголовки ответа;
- POST – позволяет загрузить информацию на сервер, по смыслу изменяет ресурс на сервере, но зачастую используется и для создания ресурса на сервере, тело запроса содержит изменяемый/создаваемый ресурс;
- PUT – аналогичен POST, но по смыслу занимается созданием ресурса, а не его изменением, тело запроса содержит создаваемый ресурс;
- DELETE – удаляет ресурс с сервера.

Кроме указанных методов HTTP, существует ещё большое количество других методов, определённых в спецификации протокола HTTP. Однако, несмотря на это, браузерами зачастую используются только методы GET и POST. Тем не менее, другие прикладные приложения могут использовать HTTP-методы по своему усмотрению.

Как мы увидели ранее, в составе HTTP-ответа содержится статусный код или код возврата. Этот статус показывает состояние HTTP-ответа, которое получено от сервера. Этот механизм является необходимым при функционировании протокола HTTP, поскольку при обработке запроса могут встречаться различные нестандартные ситуации. Все статусные коды являются трёхзначными числами. Кроме того, в составе HTTP-ответа может присутствовать текстовое описание состояния. Все статусные коды делятся на пять групп.

Каждая группа статусных кодов идентифицирует ситуацию, в которой оказался запрос. Группа определяется первым разрядом статусного кода.

Например, статусные коды группы 2xx говорят об успехе выполнения HTTP-запроса. Наиболее используемые статусные коды приведены в таблице I.5.

Таблица I.5. Коды HTTP-ответа

Код	Описание
1xx	Информационные коды
2xx	Успешное выполнение запроса
200	Запрос был обработан успешно
201	Объект создан
202	Информация принята
203	Информация, которая не заслуживает доверия
204	Нет содержимого
205	Сбросить содержимое
206	Частичное содержимое (например, при «докачке» файлов)
3xx	Перенаправление (чтобы выполнить запрос, нужны какие-либо действия)
300	Несколько вариантов на выбор
301	Ресурс перемещён на постоянной основе
302	Ресурс перемещён временно
303	Смотрите другой ресурс
304	Содержимое не изменилось
305	Используйте прокси-сервер
4xx	Проблема связана не с сервером, а с запросом
400	Некорректный запрос
401	Нет разрешения на просмотр ресурса

402	Требуется оплата
403	Доступ запрещён
404	Ресурс не найден
405	Недопустимый метод
406	Неприемлемый запрос
407	Необходима регистрация на прокси-сервере
408	Время обработки запроса истекло
409	Конфликт
410	Ресурса больше нет
411	Необходимо указать длину
412	Не выполнено предварительное условие
413	Запрашиваемый элемент слишком велик
414	Идентификатор ресурса (URI) слишком длинный
415	Неподдерживаемый тип ресурса
5xx	Ошибки на сервере
500	Внутренняя ошибка сервера
501	Функция не реализована
502	Дефект шлюза
503	Служба недоступна
504	Время прохождения через шлюз истекло
505	Неподдерживаемая версия HTTP

Эти и другие статусные коды используются для передачи информации о статусе запроса от клиента к серверу.

В ответе используется такая же структура полей заголовка, как и в запросе клиента. Поля заголовка предназначены для того, чтобы уточнить ответ сервера клиенту. Описание некоторых из полей, которые можно встретить в заголовке ответа сервера, приведено в таблице I.6.

Таблица I.6. Поля заголовка ответа web-сервера

Имя поля	Описание содержимого
Server	Имя и номер версии сервера
Age	Время в секундах, прошедшее с момента создания ресурса
Allow	Список методов, допустимых для данного ресурса
Content-Language	Языки, которые должен поддерживать клиент для того, чтобы корректно отобразить передаваемый ресурс
Content-Type	MIME-тип данных, содержащихся в теле ответа сервера
Content-Length	Число символов, содержащихся в теле ответа сервера
Last-Modified	Дата и время последнего изменения ресурса
Date	Дата и время, определяющие момент генерации ответа
Expires	Дата и время, определяющие момент, после которого информация, переданная клиенту, считается устаревшей
Location	В этом поле указывается реальное расположение ресурса. Оно используется для перенаправления запроса
Cache-Control	Директивы управления кэшированием. Например, no - cache означает, что данные не должны кэшироваться

В теле ответа содержится код ресурса, передаваемого клиенту в ответ на запрос. Это не обязательно должен быть HTML-текст web-страницы. В со-

ставе ответа могут передаваться изображение, аудиофайл, фрагмент видеоинформации, а также любой другой тип данных, поддерживаемых клиентом. О том, как следует обрабатывать полученный ресурс, клиенту сообщает содержимое поля заголовка Content-type.

Ниже представлен пример ответа сервера на запрос, приведённый в предыдущем разделе. В теле ответа содержится исходный текст HTML-документа.

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.1
X-Powered-By: ASP.NET
Date: Mon, 20 OCT 2008 11:25:56 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Thu, 4 Apr 2019 15:05:44 GMT
ETag: "b66a667f948c92:8a5"
Content-Length: 426

<html>
<body>
<form action='http://localhost/Scripts/test.pl'>
<p>Operand1: <input type='text' name='A'></p>
<p>Operand2: <input type='text' name='B'></p>
<p>Operation:<br>
<select name='op'>
<option value='+'>+</option>
<option value='-'>-</option>
<option value='*'>*</option>
<option value='/'>/</option>
<select></p>
<input type='submit' value='Calculate!'>
</from>
</body>
</html>
```

Поля заголовка и тело сообщения могут отсутствовать, но строка состояния является обязательным элементом, так как указывает на тип запроса/ответа.

Поле с именем Content-type может встречаться как в запросе клиента, так и в ответе сервера. В качестве значения этого поля указывается MIME – тип со-

держимого запроса или ответа. MIME – тип также передаётся в поле заголовка Accept, присутствующего в запросе.

Спецификация MIME (Multipurpose Internet Mail Extension – многоцелевое почтовое расширение Internet) первоначально была разработана для того, чтобы обеспечить передачу различных форматов данных в составе электронных писем. Однако применение MIME не исчерпывается электронной почтой. Средства MIME успешно используются в WWW и, по сути, стали неотъемлемой частью этой системы.

Стандарт MIME разработан как расширяемая спецификация, в которой подразумевается, что число типов данных будет расти по мере развития форм представления данных. Каждый новый тип в обязательном порядке должен быть зарегистрирован в IANA (Internet Assigned Numbers Authority).

До появления MIME компьютеры, взаимодействующие по протоколу HTTP, обменивались исключительно текстовой информацией. Для передачи изображений, как и для передачи любых других двоичных файлов, приходилось пользоваться протоколом FTP.

В соответствии со спецификацией MIME, для описания формата данных используются тип и подтип. Тип определяет, к какому классу относится формат содержимого HTTP-запроса или HTTP-ответа. Подтип уточняет формат. Тип и подтип отделяются друг от друга косой чертой: тип/подтип.

Поскольку в подавляющем большинстве случаев в ответ на запрос клиента сервер возвращает исходный текст HTML-документа, то в поле Content-type ответа обычно содержится значение text/html. Здесь идентификатор text описывает тип, сообщая, что клиенту передаётся символьная информация, а идентификатор html описывает подтип, т.е. указывает на то, что последовательность символов, содержащаяся в теле ответа, представляет собой описание документа на языке HTML.

Перечень типов и подтипов MIME достаточно велик. В таблице I.7. приведены примеры MIME – типов, наиболее часто встречающиеся в заголовках HTML-запросов и ответов.

Таблица I.7. MIME типы данных

Тип/подтип	**Расширение файла**	**Описание**
application/pdf	.pdf	Документ, предназначенный для обработки Acrobat Reader
application/msexcel	.xls	Документ в формате Microsoft Excel
application/postscript	.ps, .eps	Документ в формате PostScript
application/x-tex	.tex	Документ в формате TeX
application/msword	.doc	Документ в формате Microsoft Word
application/rtf	.rtf	Документ в формате RTF, отображаемый с помощью Microsoft Word
image/gif	.gif	Изображение в формате GIF
image/ jpeg	.jpeg, .jpg,	Изображение в формате JPEG
image/tiff	.tiff, .tif	Изображение в формате TIFF
image/x-xbitmap	.xbm	Изображение в формате XBitmap
text/plain	.txt	ASCII-текст
text/html	.html, .htm	Документ в формате HTML
audio/midi	.midi, .mid	Аудиофайл в формате MIDI
audio/x-wav	.wav	Аудиофайл в формате WAV
message/rfc822		Почтовое сообщение
message/news		Сообщение в группы новостей
video /mpeg	.mpeg, .mpg, .mpe	Видеофрагмент в формате MPEG
video/avi	.avi	Видеофрагмент в формате AVI

Отличительной особенностью протокола HTTP является то, что в рамках этого протокола информация передаётся в виде текста. Это означает, что работать с таким протоколом достаточно просто. Кроме того, инженеры по безопасности даже при строгом режиме безопасности оставляют открытым именно протокол HTTP. Поэтому реализация сетевого взаимодействия в рамках протокола HTTP является одним из перспективных направлений.

Однако, несмотря на простоту протокола, существует проблема утечки передаваемой информации. Поскольку информация передаётся в виде обычного текста, то перехват такой информации осуществляется достаточно просто. В некоторых ситуациях эта проблема не является критичной. Однако, для web-приложений, работающих с конфиденциальной информацией это достаточно существенный недостаток.

По этой причине существует модификация этого протокола – HTTPS, т.е. протокол HTTP с поддержкой шифрования.

Как известно, существуют классические криптостойкие алгоритмы шифрования, которые шифруют данные на основе существующего ключа. Для шифрования и расшифровки данных используется один и тот же ключ – если кто-либо знает ключ к зашифрованной информации, то он может расшифровать её. Ключ – это обычная последовательность бит определённой длины. Чем больше длина ключа, тем сложнее взломать алгоритм шифрования. Таким образом, для того, чтобы защитить свою информацию, необходимо хранить в секрете ключ шифрования. В этом случае используют дополнительно другой вид шифрования – ассиметричный. В этом случае существует пара ключей – открытый и закрытый. С помощью открытого ключа можно только зашифровать информацию, а с помощью закрытого – расшифровать. Обычно при таком подходе закрытый ключ хранится в секрете, а открытый ключ является общедоступным. Ассиметричный алгоритм работает медленнее, чем симметричный, поэтому его используют для первоначального обмена симметричными ключами (рис. I.4.). Давайте рассмотрим весь алгоритм работы зашифрованного соединения по HTTP.

При обращении клиента к серверу по защищённому каналу сервер хранит открытый и закрытый ключ. В начальный момент времени сервер передаёт клиенту открытый ключ ассиметричного шифрования. Клиент случайным образом генерирует ключ симметричного шифрования и шифрует его с помощью открытого ключа, полученного от сервера. После этого клиент отправляет зашифрованный ключ на сервер и в этот момент времени клиент и сервер имеют одинаковые ключи для симметричного шифрования. Далее идет HTTP-взаимодействие, которое шифруется с помощью этого симметричного ключа. Симметричный ключ остаётся в секрете и не может быть перехвачен, поскольку закрытый ключ (которым можно расшифровать первое сообщение, содержащее симметричный ключ) остаётся в секрете на сервере. Таким образом, обеспечивается конфиденциальность и целостность передаваемых данных по протоколу HTTP.

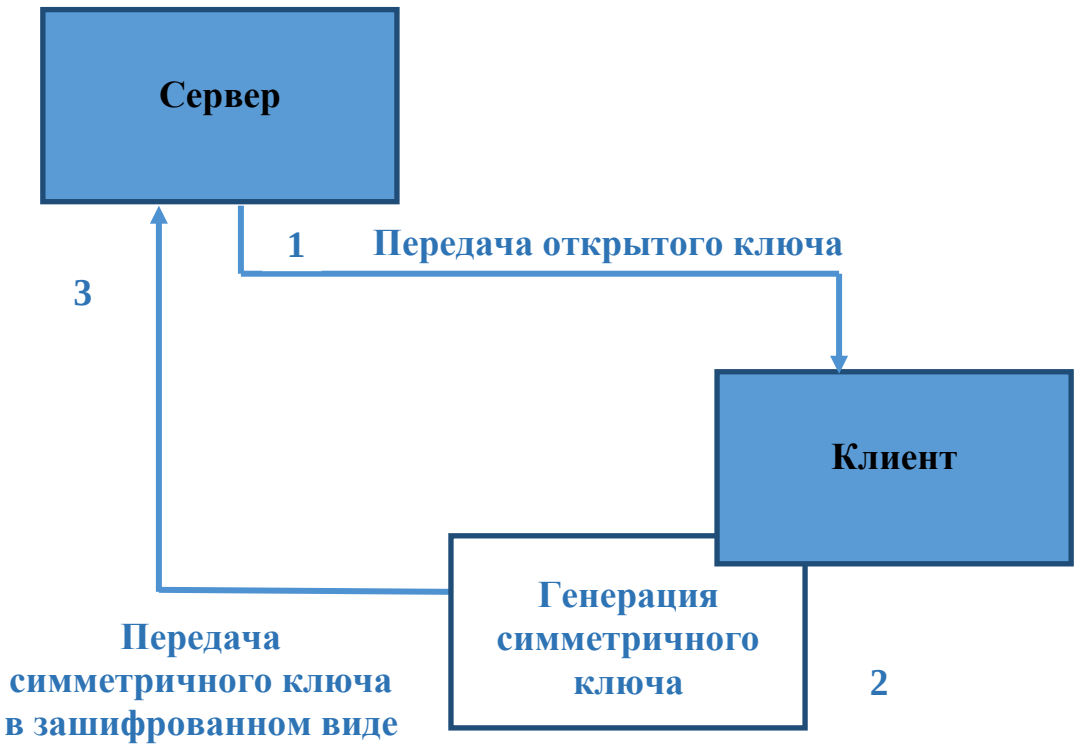


Рис. I.4. HTTP-взаимодействие

Для однозначной идентификации ресурсов в web-сети используются уникальные идентификаторы URL.

Единообразный идентификатор ресурса URI (Uniform Resource Identifier) представляет собой короткую последовательность символов, идентифицирующую абстрактный или физический ресурс. URI не указывает на то, как получить ресурс, а только идентифицирует его. Это даёт возможность описывать с помощью RDF (Resource Description Framework) ресурсы, которые не могут быть получены через интернет (имена, названия и т.п.). Самые известные примеры URI – это URL и URN.

- URL (Uniform Resource Locator) – это URI, который, помимо идентификации ресурса, предоставляет ещё и информацию о местонахождении этого ресурса
- URN (Uniform Resource Name) – это URI, который идентифицирует ресурс в определённом пространстве имён, но, в отличие от URL, URN не указывает на местонахождение этого ресурса.

URL имеет следующую структуру:

<схема>://<логин>:<пароль>@<хост>:<порт>/<URL-путь>, где:

- схема – схема обращения к ресурсу (обычно сетевой протокол);
- логин – имя пользователя, используемое для доступа к ресурсу;
- пароль – пароль, ассоциированный с указанным именем пользователя;
- хост – полностью прописанное доменное имя хоста в системе DNS или IP-адрес хоста;
- порт – порт хоста для подключения;
- URL-путь – уточняющая информация о месте нахождения ресурса.

Общепринятые схемы (протоколы) URL включают протоколы: ftp, http, https, telnet, а также:

- gopher – протокол Gopher;
- mailto – адрес электронной почты;
- news – новости Usenet;

- nntp – новости Usenet через протокол NNTP;
- irc – протокол IRC;
- prospero – служба каталогов Prospero Directory Service;
- wais – база данных системы WAIS;
- xmpp – протокол XMPP (часть Jabber);
- file – имя локального файла;
- data – непосредственные данные (Data: URL).

I.5. Обеспечение безопасности передачи данных HTTP

Поскольку протокол HTTP предназначен для передачи символьных данных в открытом (незашифрованном) виде, то лица, имеющие доступ к каналу передачи данных между клиентом и сервером, могут без труда просматривать весь трафик и использовать его для совершения несанкционированных действий. В связи с этим предложен ряд расширений базового протокола, направленных на повышение защищённости интернет-трафика от несанкционированного доступа.

Самым простейшим является расширение HTTPS, при котором данные, передаваемые по протоколу HTTP, «упаковываются» в криптографический протокол SSL или TLS, тем самым обеспечивая защиту этих данных. В отличие от HTTP, для HTTPS по умолчанию используется TCP-порт 443. Чтобы подготовить web-сервер для обработки HTTPS соединений, администратор должен получить и установить в систему сертификат для этого web-сервера.

SSL (Secure Sockets Layer) – криптографический протокол, обеспечивающий безопасную передачу данных по сети интернет. При его использовании создаётся защищённое соединение между клиентом и сервером. SSL изначально разработан компанией Netscape Communications. Впоследствии на основании протокола SSL 3.0 был разработан и принят стандарт RFC, получивший название TLS. Этот протокол использует шифрование с открытым ключом для подтверждения подлинности передатчика и получателя. Поддерживает надёжность передачи данных за счёт использования корректирующих кодов и без-

опасных хеш-функций. На нижнем уровне многоуровневого транспортного протокола (например, TCP) он является протоколом записи и используется для инкапсуляции различных протоколов (POP3, IMAP, SMTP или HTTP). Для каждого инкапсулированного протокола он обеспечивает условия, при которых сервер и клиент могут подтверждать друг другу свою подлинность, выполнять алгоритмы шифрования и производить обмен криптографическими ключами, прежде чем протокол прикладной программы начнёт передавать и получать данные.

Для доступа к web-страницам, защищённым протоколом SSL, в URL вместо схемы http, как правило, подставляется схема https, указывающая на то, что будет использоваться SSL-соединение. Стандартный TCP-порт для соединения по протоколу https – 443. Для работы SSL требуется, чтобы на сервере имелся SSL-сертификат.

В web-сети поддерживаются 3 типа аутентификации при клиент-серверных взаимодействиях:

- Basic – базовая аутентификация, при которой имя пользователя и пароль передаются в заголовках http-пакетов. Пароль при этом не шифруется и присутствует в чистом виде в кодировке base64. Для данного типа аутентификации использование SSL является обязательным.
- Digest – дайджест-аутентификация, при которой пароль пользователя передаётся в хешированном виде. По уровню конфиденциальности паролей этот тип мало чем отличается от предыдущего, так как атакующему все равно, действительно ли это настоящий пароль или только хеш от него: перехватив удостоверение, он все равно получает доступ к конечной точке. Для данного типа аутентификации использование SSL является обязательным.
- Integrated – интегрированная аутентификация, при которой клиент и сервер обмениваются сообщениями для выяснения подлинности друг друга с помощью протоколов NTLM или Kerberos. Этот тип

аутентификации защищён от перехвата удостоверений пользователей, поэтому для него не требуется протокол SSL. Только при использовании данного типа аутентификации можно работать по схеме http, во всех остальных случаях необходимо использовать схему https.

Cookie

Поскольку HTTP-сервер не помнит предыстории запросов клиентов, то каждый запрос обрабатывается независимо от других, и у сервера нет возможности определить, исходят ли запросы от одного клиента или разных клиентов.

Если сервер будет проверять TCP-соединения и запоминать IP-адреса компьютеров-клиентов, он все равно не сможет различить запросы от двух браузеров, выполняющихся на одной машине. И даже если допустить, что на компьютере работает лишь одна клиент-программа, то никто не может утверждать, что в промежутке между двумя запросами она не была завершена, а затем запущена снова уже другим пользователем.

Тем не менее, если вы когда-нибудь пользовались почтовым ящиком на mail.ru или на другом сервере, предоставляющем почтовые web-услуги пользователям, вспомните, как вёл себя клиент после того, как вы создали для себя почтовый ящик на сервере. Когда вы в следующий раз обратились с того же компьютера к mail.ru, вы, вероятно, заметили, что после загрузки web-страницы ваше регистрационное имя уже отображалось в соответствующем поле ввода.

Такие сведения позволяет получить дополнительное средство под названием cookie. Механизм cookie позволяет серверу хранить информацию на компьютере клиента и извлекать её оттуда.

Инициатором записи cookie выступает сервер. Если в ответе сервера присутствует поле заголовка Set-cookie, клиент воспринимает это как команду на запись cookie. В дальнейшем, если клиент обращается к серверу, от которого он ранее принял поле заголовка Set-cookie, помимо прочей информации он передаёт серверу данные cookie. Для передачи указанной информации серверу используется поле заголовка cookie.

Для того чтобы в общих чертах представить себе, как происходит обмен данными cookie, рассмотрим следующий пример. Предположим, что клиент передаёт запросы на серверы A, B и C. Предположим также, что сервер B, в отличие от A и C, передаёт клиенту команду записать cookie. Последовательность запросов клиента серверу и ответов на них будет выглядеть приблизительно следующим образом.

1. Передача запроса серверу A.
2. Получение ответа от сервера A.
3. Передача запроса серверу B.
4. Получение ответа от сервера B. В состав ответа входит поле заголовка SetCookie. Получив его, клиент записывает cookie на диск.
5. Передача запроса серверу C. Несмотря на то, что на диске хранится запись cookie, клиент не предпринимает никаких специальных действий, так как значение cookie было записано по инициативе другого сервера.
6. Получение ответа от сервера C.
7. Передача запроса серверу A. В этом случае клиент также никак не реагирует на тот факт, что на диске хранится cookie.
8. Получение ответа от сервера A.
9. Передача запроса серверу B. Перед тем как сформировать запрос, клиент определяет, что на диске хранится запись cookie, созданная после получения ответа от сервера B. Клиент проверяет, удовлетворяет ли данный запрос некоторым требованиям, и, если проверка даёт положительный результат, включает в заголовок запроса поле cookie.

Таким образом, процедуру записи и получения cookie можно представить себе как своеобразный «запрос» сервера, инкапсулированный в его ответе клиенту. Соответственно, получение cookie также можно представить себе как ответ клиента, инкапсулированный в составе запроса тому же серверу.

Рассмотрим подробнее, какие данные передаются в поле заголовка Set-cookie и как они влияют на поведение клиента.

Поле Set-cookie имеет следующий формат:

Set-cookie: имя=значение; expires=дата; path=путь; domain= имя_домена, secure, где

- Пара имя=значение – именованные данные, сохраняемые с помощью механизма cookie. Эти данные должны храниться на клиент-машине и передаваться серверу в составе очередного запроса клиента.
- Дата, являющаяся значением параметра expires, определяет время, по истечении которого информация cookie теряет свою актуальность. Если ключевое слово expires отсутствует, данные cookie не удаляются по окончании текущего сеанса работы браузера.
- Значение параметра domain определяет домен, с которым связываются данные cookie. Чтобы узнать, следует ли передавать в составе запроса данные cookie, браузер сравнивает доменное имя сервера, к которому он собирается обратиться, с доменами, которые связаны с записями cookie, хранящимися на клиент-машине. Результат проверки будет считаться положительным, если сервер, которому направляется запрос, принадлежит домену, связанному с cookie. Если соответствие не обнаружено, данные cookie не передаются.
- Путь, указанный в качестве значения параметра path, позволяет выполнить дальнейшую проверку и принять окончательное решение о том, следует ли передавать данные cookie в составе запроса. Помимо домена с записью cookie связывается путь. Если браузер обнаружил соответствие имени домена значению параметра domain, он проверяет, соответствует ли путь к ресурсу пути, связанному с cookie. Сравнение считается успешным, если ресурс содержится в каталоге, указанном посредством ключевого слова path, или в одном из его подкаталогов. Если и эта проверка даёт положительный

результат, данные cookie передаются серверу. Если параметр path в поле Set-Cookie отсутствует, то считается, что запись cookie связана с URL конкретного ресурса, передаваемого сервером клиенту.

- Последний параметр, secure, указывает на то, что данные cookie должны передаваться по защищённому каналу.

Для передачи данных cookie серверу используется поле заголовка cookie. Формат этого поля достаточно простой:

Cookie: имя=значение; имя=значение; ...

С помощью поля Cookie передаётся одна или несколько пар имя=значение. Каждая из этих пар принадлежит записи cookie, для которой URL запрашиваемого ресурса соответствуют имени домена и пути, указанным ранее в поле Set-cookie.

Структура IP адреса

IP-адрес состоит из двух частей: номера сети и номера узла в сети.

Таблица I.8. Распределение адресов в IP сетях

Класс	Первые биты	Наименьший номер сети	Наибольший номер сети	Максимальное число узлов в сети
A	0	1.0.0.0 (0 - не используется)	126.0.0.0 (127-зарезервирован)	2^{24} (3 байта)
B	10	128.0.0.0	191.255.0.0	2^{16} (2 байта)
C	110	192.0.0.0	223.255.255.0	2^{8} (1 байт)
D	1110	224.0.0.0	239.255.255.255	групповые адреса
E	11110	240.0.0.0	247.255.255.255	зарезервировано

Самой распространённой является запись IP-адреса в виде четырёх чисел, разделённых точками, каждое из которых представляет значение байта в деся-

тичной форме, например: 213.180.204.11. Запись адреса не предусматривает специального разграничительного знака между номером сети и номером узла.

Для разделения этих частей обычно используется 2 подхода:

1. С помощью маски (RFC 950, RFC 1518), представляющей собой число в паре с IP-адресом. С помощью операции «логическое И» над этими двумя числами выделяется номер сети.
2. С помощью классов адресов (RFC 791).

Вводится пять классов адресов: A, B, C, D, E.

A, B, C – используются для адресации сетей, D и E – имеют специальное назначение. Признаком, на основании которого IP-адрес относят к тому или иному классу, являются значения нескольких первых битов адреса.

В рамках IP протокола существуют ограничения при назначении IP-адресов, а именно:

- номера сетей и номера узлов не могут состоять из двоичных нулей или единиц;
- если IP-адрес состоит только из двоичных нулей, то он называется неопределённым адресом и обозначает адрес того узла, который сгенерировал этот пакет;
- если в поле номера сети стоят только нули, то по умолчанию считается, что узел назначения принадлежит той же самой сети, что и узел, который отправил пакет; такой адрес может быть использован только в качестве адреса отправителя;
- если все двоичные разряды IP-адреса равны 1, то пакет с таким адресом назначения должен рассылаться всем узлам, находящимся в той же сети, что и источник этого пакета; такой адрес называется ограниченным широковещательным, поскольку пакет не сможет выйти за границы сети;
- если в поле адреса назначения в разрядах, соответствующих номеру узла, стоят только единицы, то пакет рассылается всем узлам сети,

номер которой указан в адресе назначения; такой тип адреса называется широковещательным;

- если первый октет адреса равен 127, то такой адрес называется внутренним адресом стека протоколов, он используется для тестирования программ, организации клиентской и серверной частей приложений, установленных на одном компьютере;
- групповые адреса, относящиеся к классу D, предназначены для экономичного распространения в интернете, большой корпоративной сети аудио- или видеопрограмм.

Стандартным классам сетей можно поставить в соответствие следующие значения маски:

- класс A – 255.0.0.0;
- класс B – 255.255.0.0;
- класс C – 255.255.255.0.

Рассмотрим следующий пример:

Исходные данные:	IP адрес	62.76.167.21
	Маска сети	255.255.255.0
Логическая операция:	И	
Результат:	Адрес сети	62.76.167.0
	Номер компьютера	21

Для определения сетевых настроек компьютера и сетевого оборудования, диагностики и получения другой информации, относящейся к интернет-протоколам, широко используются специальные утилиты.

Утилита ipconfig

Ipconfig – это утилита командной строки для вывода деталей текущего соединения компьютера с сетью и контроля над клиентским сервисом DHCP. DHCP (Dynamic Host Configuration Protocol) – это сетевой протокол,

позволяющий компьютерам автоматически получать IP-адрес и другие параметры, необходимые для работы в сети TCP/IP.

Синтаксис команды: ipconfig/ключи.

Команда ipconfig/all – отображает полную информацию по всем сетевым адаптерам.

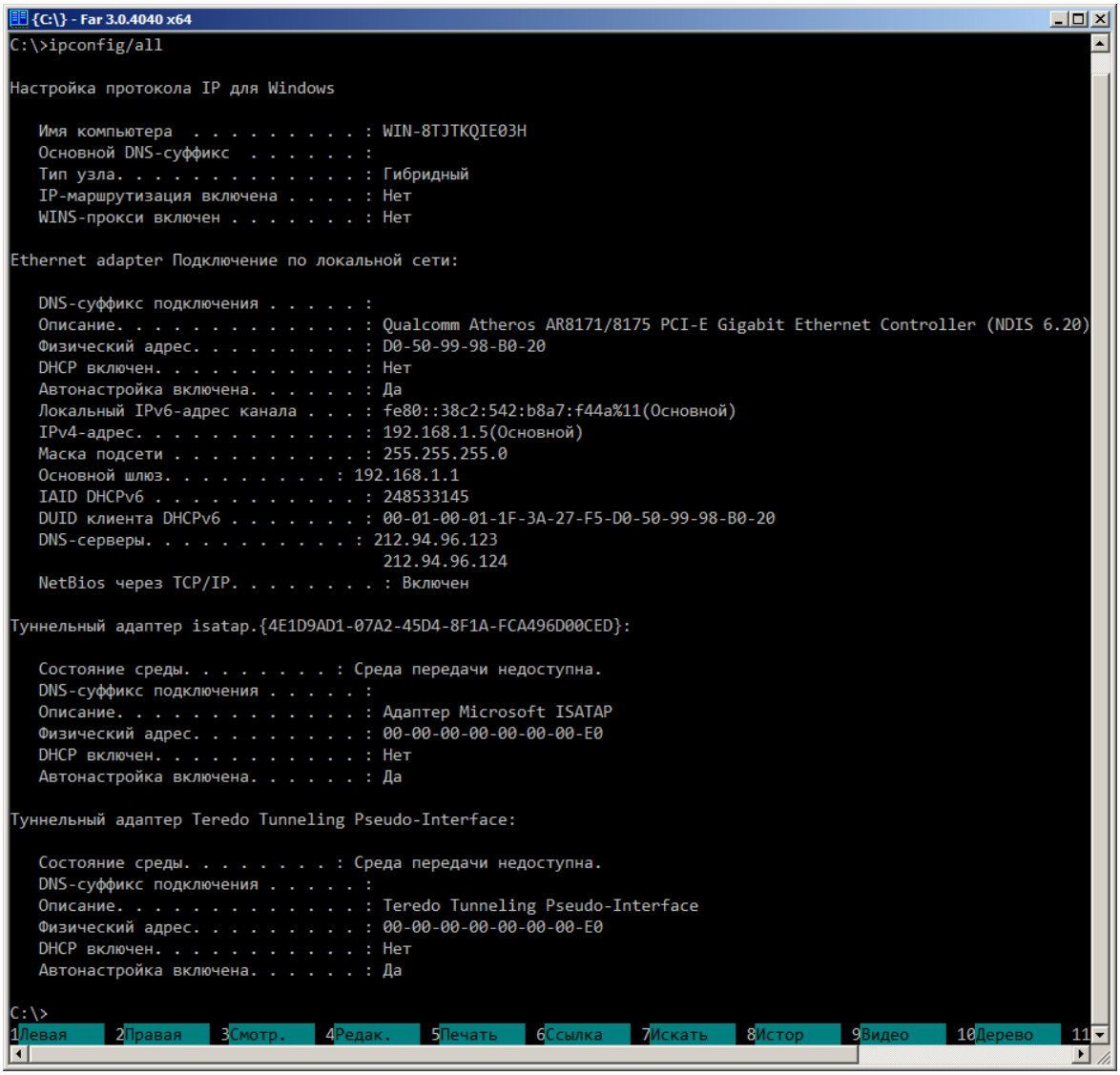


Рис. I.5. Пример вывода команды ipconfig/all для Windows

Утилита ping

Ping (Packet InterNet Grouper) – это системная программа, предназначенная для проверки соединений в сетях на основе TCP/IP. Она отправляет Echo-Request запросы протокола ICMP указанному узлу сети и фиксирует поступающие ответы (ICMP Echo-Reply). Время между отправкой запроса и получени-

ем ответа (RTT, Round Trip Time) позволяет определять двусторонние задержки (RTT) по маршруту и частоту потери пакетов. Что позволяет косвенно определять загруженность каналов передачи данных и промежуточных устройств. Полное отсутствие ICMP-ответов может также означать, что удалённый узел (или какой-либо из промежуточных маршрутизаторов) блокирует ICMP Echo-Reply или игнорирует ICMP Echo-Request.

Синтаксис: ping – параметры конечное_имя, где конечное имя – это доменное имя или IP-адрес хоста

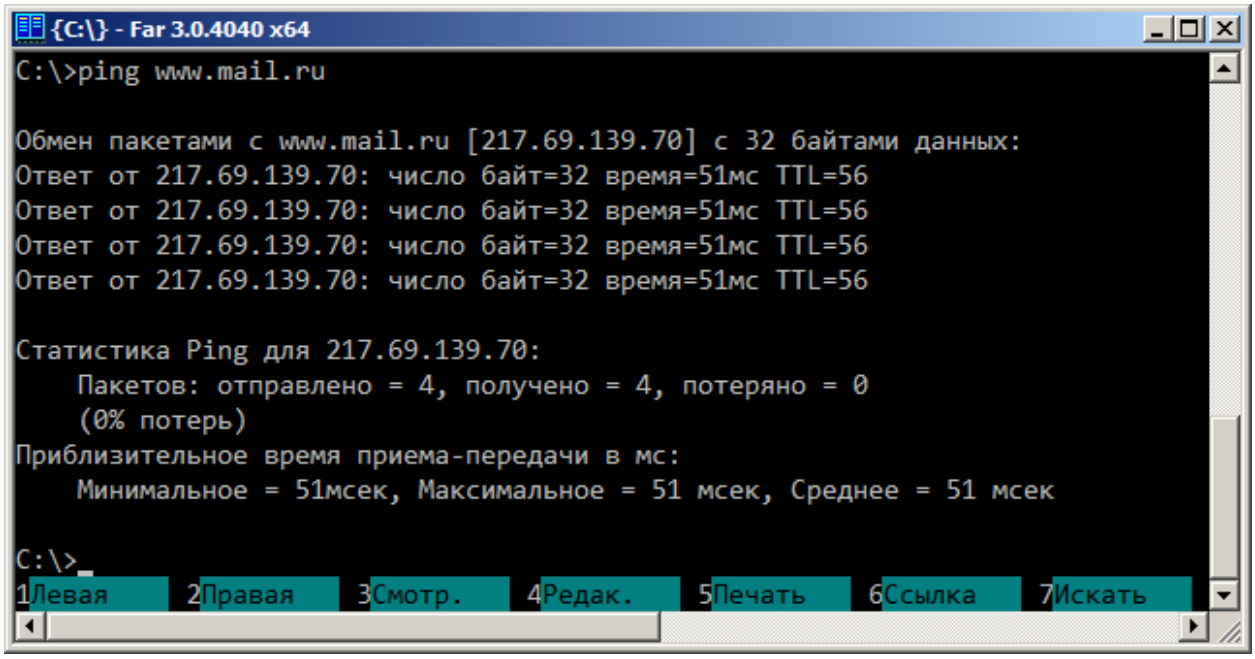


Рис. I.6. Пример вывода команды ping

Утилита traceroute

Traceroute (сокращенно tracert) – это служебная программа, предназначенная для определения маршрутов следования пакетов в сетях TCP/IP. Работа traceroute основана на протоколе ICMP.

Traceroute выполняет отправку пакетов указанному узлу сети, отображая при этом сведения о всех промежуточных маршрутизаторах, через которые прошли пакеты на пути к целевому узлу. В случае проблем при доставке пакетов до какого-либо узла программа traceroute позволяет определить, на каком именно участке сети возникли неполадки.

Синтаксис: tracert – параметры конечное_имя, где конечное_имя – это доменное имя или IP-адрес хоста.

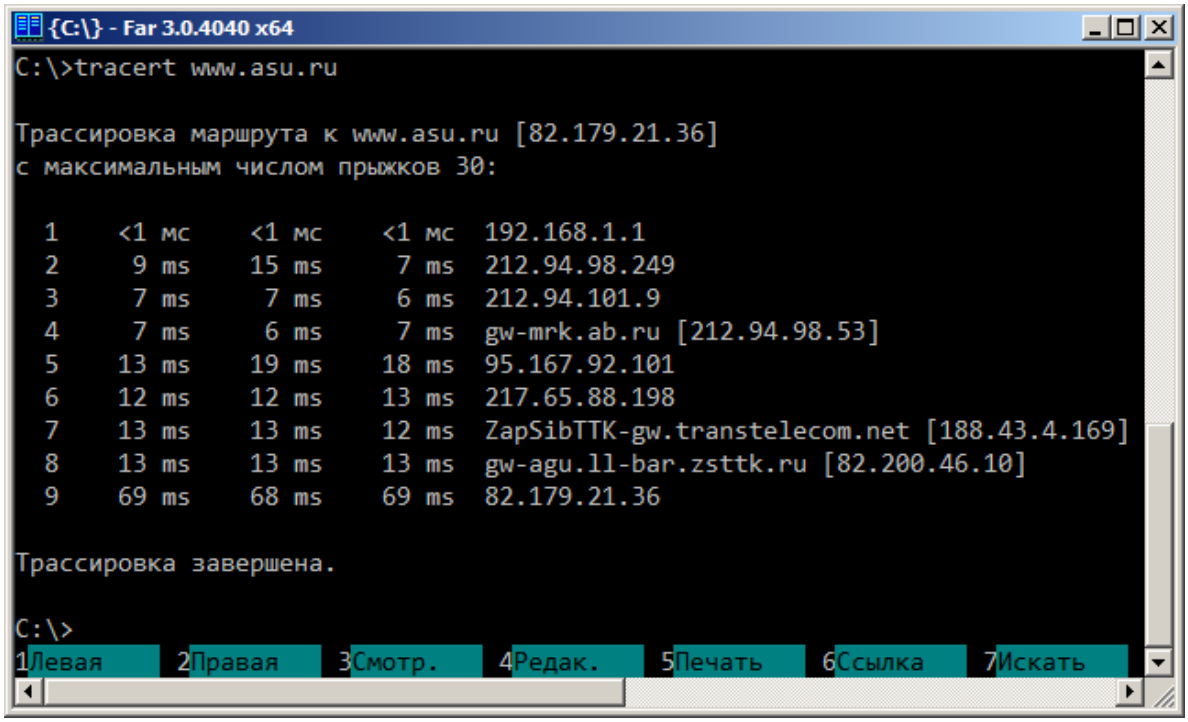


Рис. I.7. Пример вывода команды tracert

Утилита netstat

Netstat – служебная программа, отображающая статистику протокола и текущих сетевых подключений TCP/IP:

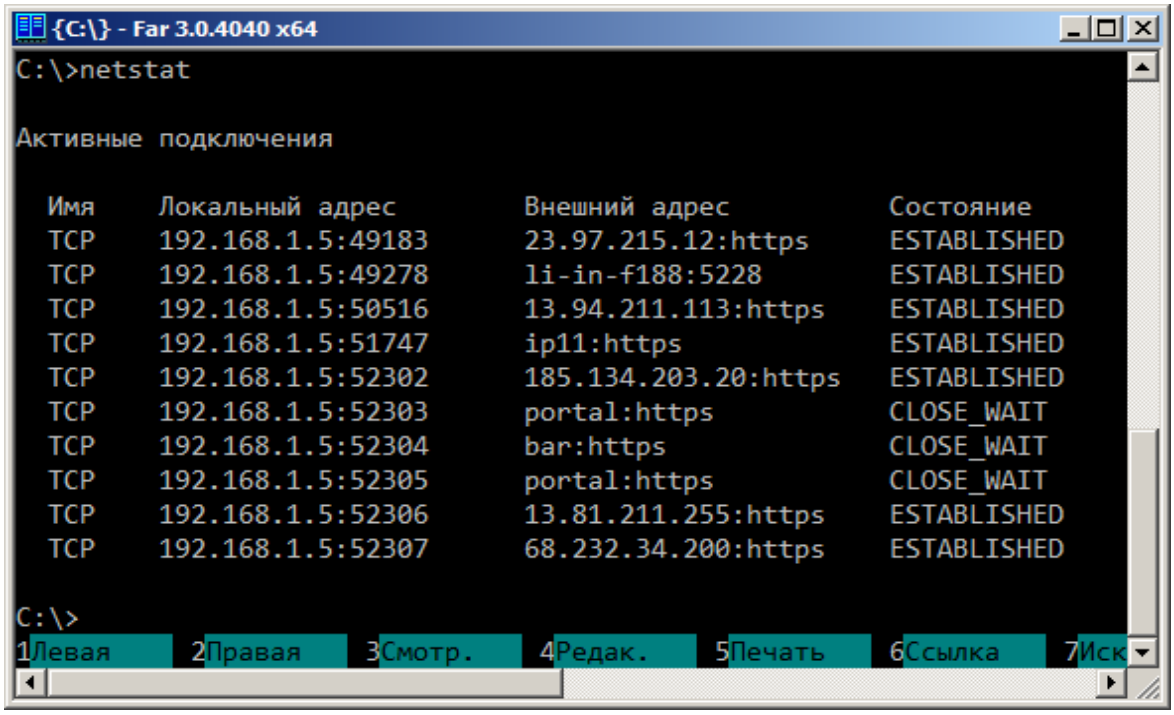


Рис. I.8. Пример вывода команды Netstat

Утилита telnet

Telnet – сетевой протокол для реализации текстового интерфейса по сети. Название «telnet» имеет также утилита, реализующая клиентскую часть протокола. Исторически telnet служил для удалённого доступа к интерфейсу командной строки операционных систем. Протокол telnet может использоваться для выполнения отладки других протоколов на основе транспорта TCP.

Утилита telnet поддерживает следующие команды:

- Close – закрытие текущего подключения.
- Display – отображение параметров операции.
- Open – подключение к сайту.
- Quit – выход из telnet.
- Set – установление параметров.
- Send – отправление строки на сервер.
- Status – вывод сведений о текущем состоянии.
- Unset – сброс параметров.

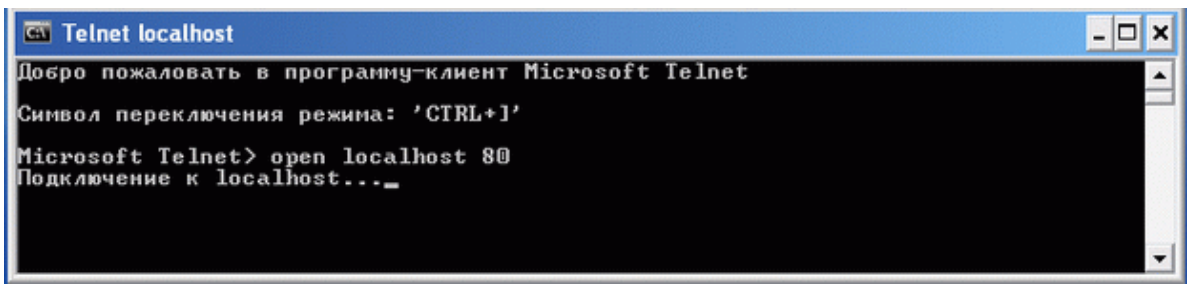
Используя утилиту telnet, можно, например вручную, отправить запрос клиента и получить ответ сервера по протоколу HTTP.

Для этого выполним следующую последовательность действий:

1. Запуск утилиты telnet.
2. Установление соединения с web-сервером с помощью команды: open имя_хоста 80.
3. Формирование запроса клиента.
4. Получение ответа сервера.

Пример:

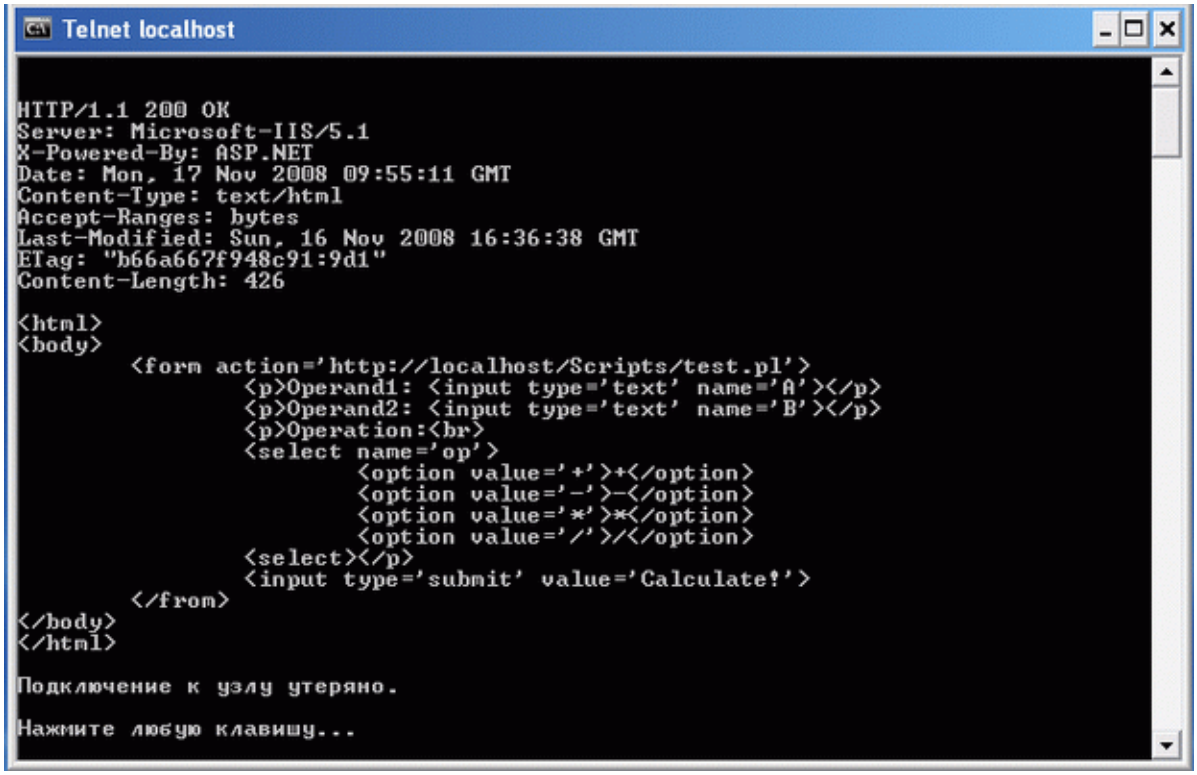
1. Устанавливаем соединение: open localhost 80.



2. Формируем строку состояния запроса клиента:

GET http://localhost/perlcalc.html http/1.0 <enter>

3. Получаем ответ сервера:



Видно, что ответ web-сервера localhost содержит строку состояния (с кодом успешного завершения 200), поля заголовка (Server, Date, Content-type и др.) и тело, содержащее HTML код запрошенного клиентом документа http://localhost/perlcalc.html.

II. Уязвимости по классификации OWASP

Уязвимости web-приложений – это возможность их взлома из-за наличия ошибок в программном коде web-приложений, неправильных настроек системы управления или программного обеспечения сервера. Уязвимости в web-приложениях являются одним из наиболее распространённых недостатков обеспечения защиты информации. Поэтому обеспечение безопасности web-приложений является важным элементом защиты информационных систем.

Проект Open Web Application Security ™ (или сокращенно OWASP) – это открытый проект безопасности web-приложений. OWASP является всемирной некоммерческой благотворительной организацией, которая занимается вопросами повышения безопасности программного обеспечения и осведомлённости специалистов отрасли информационной безопасности в вопросах разработки, эксплуатации и защиты web-приложений.

Работая как сообщество профессионалов-единомышленников, OWASP выпускает программные инструменты и документацию по безопасности web-приложений, доступные бесплатно в открытом доступе:

- стандарты и инструменты для обеспечения безопасности приложений;
- полные версии книг по тестированию безопасности приложений, разработке безопасного кода, а также оценке безопасности кода;
- презентации и видео;
- памятки по большинству распространенных вопросов;
- стандартные требования к безопасности и библиотеки;
- локальные отделения по всему миру;
- передовые исследования;
- крупные конференции по всему миру;
- списки рассылок.

Одним из наиболее популярных и востребованных документов, опубликованных OWASP, является список OWASP Top-10, переведённый на многие языки мира.

OWASP Top-10 – это список десяти наиболее опасных рисков информационной безопасности для web-приложений по мнению экспертного сообщества OWASP на момент выпуска рейтинга. Для каждого пункта рейтинга риск посчитан экспертами на основе методики OWASP Risk Rating Methodology и включает оценку по следующим критериям:

- распространенность соответствующих уязвимостей в приложениях (Weakness Prevalence);
- сложность обнаружения (Weakness Detectability);
- эксплуатация (Exploitability);
- критичность (ущерб) последствий эксплуатации (Technical Impacts).

Первая версия рейтинга OWASP Top-10 опубликована в 2004 году, и с тех пор документ обновляется каждые три-четыре года. Обновлённые версии публиковались в 2007, 2010, 2013 и 2017 годах соответственно.

Последняя версия рейтинга, опубликованная в 2017 году, основана главным образом на 40+ комплектах данных, полученных от организаций, которые специализируются на безопасности приложений, а также на отраслевых исследованиях, проведённых более 500 независимыми исследователями, данные содержат информацию об уязвимостях, обнаруженных в сотнях организаций и более 100 000 реальных приложений и API.

OWASP Top-10 2017 содержит следующие риски:

- A1: Внедрение (Injection). Уязвимости, связанные, например, с внедрением SQL, NoSQL, OS и LDAP, возникают, когда непроверенные данные отправляются интерпретатору в составе команды или запроса. Вредоносные данные могут заставить интерпретатор выполнить непредусмотренные команды или обратиться к данным без прохождения соответствующей авторизации.

- A2: Недостатки аутентификации (Broken Authentication). Функции приложений, связанные с аутентификацией и управлением сессиями, часто некорректно реализуются, позволяя злоумышленникам скомпрометировать пароли, ключи или сессионные токены, а также эксплуатировать другие ошибки реализации для временного или постоянного перехвата учётных записей пользователей.
- A3: Разглашение конфиденциальных данных (Sensitive Data Exposure). Многие web-приложения и API имеют плохую защиту критичных финансовых, медицинских или персональных данных. Злоумышленники могут похитить или изменить эти данные, а затем осуществить мошеннические действия с кредитными картами или персональными данными. Конфиденциальные данные требуют дополнительных мер защиты, например, их шифрования при хранении или передаче, а также специальных мер предосторожности при работе с браузером.
- A4: Внешние сущности XML (XXE) (XXE – XML External Entities). Старые или плохо настроенные XML-процессоры обрабатывают ссылки на внешние сущности внутри документов. Эти сущности могут быть использованы для доступа к внутренним файлам через обработчики URI файлов, общие папки, сканирование портов, удалённое выполнения кода и отказ в обслуживании.
- A5: Недостатки контроля доступа (Broken Access Control). Действия, разрешённые аутентифицированным пользователям, зачастую некорректно контролируются. Злоумышленники могут воспользоваться этими недостатками и получить несанкционированный доступ к учётным записям других пользователей или конфиденциальной информации, а также изменить пользовательские данные или права доступа.
- A6: Некорректная настройка параметров безопасности (Security Misconfiguration). Некорректная настройка безопасности является

распространённой ошибкой. Это происходит из-за использования стандартных параметров безопасности, неполной или специфичной настройки, открытого облачного хранения, некорректных HTTP-заголовков и подробных сообщений об ошибках, содержащих критичные данные. Все ОС, фреймворки, библиотеки и приложения должны быть не только настроены должным образом, но и своевременно корректироваться и обновляться.

- A7: Межсайтовое выполнение сценариев (XSS) (XSS – Cross-Site Scripting). XSS имеет место, когда приложение добавляет непроверенные данные на новую web-страницу без их соответствующей проверки или преобразования, или когда обновляет открытую страницу через API браузера, используя предоставленные пользователем данные, содержащие HTML- или JavaScript-код. С помощью XSS злоумышленники могут выполнять сценарии в браузере жертвы, позволяющие им перехватывать пользовательские сессии, подменять страницы сайта или перенаправлять пользователей на вредоносные сайты.
- A8: Небезопасная десериализация (Insecure Deserialization). Небезопасная десериализация часто приводит к удалённому выполнению кода. Ошибки десериализации, не приводящие к удалённому выполнению кода, могут быть использованы для атак с повторным воспроизведением, внедрением и повышением привилегий.
- A9: Использование компонентов с известными уязвимостями (Using Components with Known Vulnerabilities). Компоненты, такие как библиотеки, фреймворки и программные модули, запускаются с привилегиями приложения. Эксплуатация уязвимого компонента может привести к потере данных или перехвату контроля над сервером. Использование приложениями и API компонентов с известными уязвимостями может нарушить защиту приложения и привести к серьёзным последствиям.

- A10: Недостатки журналирования и мониторинга (Insufficient Logging & Monitoring). Недостатки журналирования и мониторинга, а также отсутствие или неэффективное использование системы реагирования на инциденты, позволяет злоумышленникам развить атаку, скрыть своё присутствие и проникнуть в другие системы, а также изменить, извлечь или уничтожить данные. Проникновение в систему обычно обнаруживают только через 200 дней и, как правило, сторонние исследователи, а не в рамках внутренних проверок или мониторинга.

Для каждого пункта в OWASP Top-10 представлена общая информация, описание, рекомендации по предотвращению, а также примеры соответствующих атак и полезные ссылки, базовые способы защиты от подобных рисков и руководства по дальнейшим действиям.

Таким образом, основной целью OWASP Top-10 является ознакомление разработчиков, проектировщиков, архитекторов, менеджеров и организаций в целом с рисками, связанными с наиболее распространёнными и существенными недостатками в безопасности web-приложений.

Некоторые позиционируют данный рейтинг как «лучшую практику защиты web-приложений», однако такая позиция не совсем верна. Во-первых, рейтинг не покрывает все возможные уязвимости и поэтому не является достаточным источником для полной защиты web-приложений. Даже в самом документе OWASP, описывающем Top-10, есть призыв: «Не останавливайтесь на 10». Во-вторых, нужно помнить про его ограничения и недостатки: довольно большой период между обновлениями, неуточнённая предметная область (для тех приложений, которые используют только новые Фреймворки, большая часть рейтинга может быть неактуальна), неполные исходные данные (в основном, полученные компаниями на основе автоматических тестирований).

III. Основные уязвимости WEB-приложений и способы защиты

III.1. DDoS-атака

Одним из наиболее критичных по последствиям классом компьютерных атак являются распределённые атаки типа «отказ в обслуживании» DDoS (Distributed Denial of Service). Данные атаки направлены на нарушение доступности информационных ресурсов.

DDoS-атака – это комплекс действий, который способен частично или полностью сделать неработоспособным информационный ресурс. В качестве жертвы может выступать web-сайт, игровой сервер или государственный ресурс, то есть практически любой сетевой ресурс. В настоящее время в одиночку организовать DDoS-атаку практически невозможно. В большинстве случаев злоумышленник использует сеть из компьютеров, заражённых вирусом, который позволяет получать необходимый и достаточный удалённый доступ к заражённому компьютеру. Сеть из подобных компьютеров называется ботнет. Зачастую в ботнетах присутствует координирующий сервер, которому злоумышленник, решив реализовать атаку, отправляет команду, а сервер, в свою очередь даёт сигнал каждому боту начать выполнение вредоносных сетевых запросов (рис. III.1.).

Если не брать в расчёт крупные атаки, то можно сказать, что мощность подобных атак растёт более чем в 3-4 раза каждый год, а география злоумышленников меняется лишь отчасти, поскольку это объясняется максимальным количеством компьютеров в отдельно взятой стране. Согласно статистике прошлых лет, рекордсменами по числу ботов выступают такие страны, как Россия, США и Китай.

Последствия DDoS-атак варьируются от выхода из строя отдельных узлов и сервисов до остановки работы корневых DNS-серверов и частичного или полного прекращение функционирования сети.

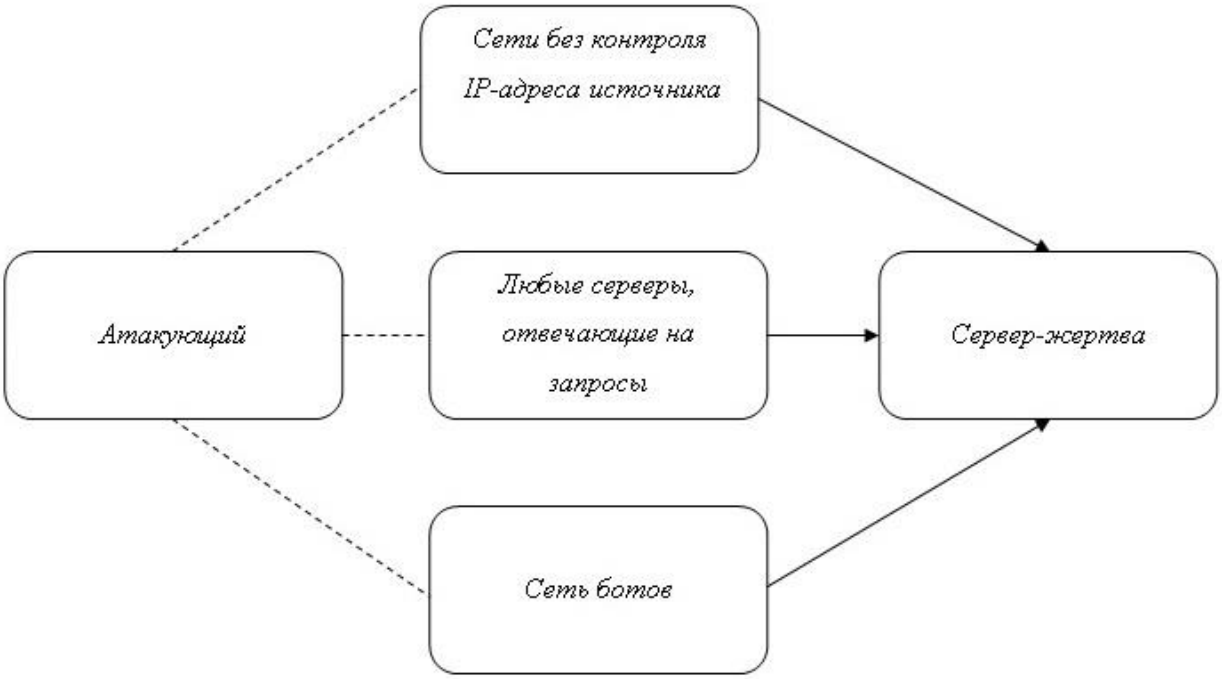


Рис. III.1. DDOS-атака

Причины DDoS-атак

DDoS-атак могут осуществляться по самым разнообразным мотивам. Ниже рассмотрим основные причины DDoS-атак:

1. Личная неприязнь.

Какое-то время назад независимый журналист-исследователь Брайан Кребс раскрыл деятельность крупного сервиса по осуществлению заказных DDoS-атак – vDOS. Информация была освещена во всех подробностях, что повлекло арест организаторов данного сервиса. В ответ хакеры организовали атаку на блог журналиста, которая стала самой крупной в мире за все годы, поскольку её мощность достигала 1 Тбит/с.

2. Развлечение.

Сегодня становится легче и проще организовать простейшую DDoS-атаку своими силами. Но подобная атака будет не совершенна и не анонимна абсолютно. Большинство из тех, кто решил попробовать себя в роли злоумышленника, не догадываются ни о первом, ни о втором. Тем не менее, многие школь-

ники часто практикуют DDoS-атаки. Итоги таких случаев бывают самыми разнообразными, вплоть до ареста.

3. Политический протест (так называемый хактивизм).

В 1996 году хакером Omega была организована и реализована одна из первых атак DDoS-атак, имеющих политическую и социальную почву. Злоумышленник являлся членом хакерской коалиции «Cult of the Dead Crew» (cDc). В связи с участившимися кибератаками на социальной почве термин «хактивизм» стал популярным в средствах массовой информации.

4. Недобросовестная конкуренция.

Такие мотивы нередки в торговой отрасли, но чаще случаются в индустрии игровых серверов. Этот довольно действенный способ недобросовестной конкуренции способен разрушить репутацию торговой компании или площадки, если её владельцы вовремя не обратятся за помощью к специалистам. Эту причину DDoS-атак можно выделить, как часто встречающуюся.

5. Вымогательство или шантаж.

В данном случае злоумышленник требует денежную сумму либо за прекращение атаки, либо с потенциальной жертвы за несовершение атаки. Зачастую подвергаются таким атакам крупные организации, например, в течение 2014 были атакованы банк «Тинькофф» и IT-ресурс Хабрахабр, крупнейший торрент-трекер Rutracker.org.

Виды DDoS-атак

Виды атак можно разделить на 3 класса:

1. Атаки, направленные на переполнение канала.

К этому виду атак можно отнести DNS амплификацию, ICMP флуд, UDP флуд и другие атаки с амплификацией, суть которых заключается в усилении атак с использованием сторонних серверов сети Интернет.

2. Атаки, использующие уязвимости стека сетевых протоколов.

Представителями атак этого вида являются ACK/PUSH ACK флуд, SYN флуд, TCP null/IP null атака, Пинг смерти (Pingofdeath);

3. Атаки на уровень приложений.

Здесь можно выделить следующие виды: HTTP флуд, атака фрагментированными HTTP пакетами, атака медленными сессиями (SlowLoris).

Способы воздействия на DDoS-трафик

- HTTP-запросы.

Самым простым видом трафика являются HTTP-запросы, с помощью которых, например, любой посетитель общается с определённым сайтом посредством браузера. В основе запроса лежит HTTP-заголовок.

- HTTP-заголовок.

HTTP-заголовки – это поля, которые описывают, какой именно ресурс запрашивается, например, URL-адрес. Кроме того, HTTP-заголовки информируют web-сервер о типе используемого браузера. К наиболее распространённым HTTP-заголовкам относят accept, language и useragent. Запрашивающая сторона может использовать любое число заголовков и придавать им нужные свойства. Злоумышленники же при проведении DDoS-атаки могут изменять HTTP-заголовки, таким образом делая их труднораспознаваемыми для выявления атаки.

- HTTP(S) GET-флуд.

HTTP(S) GET-запрос – метод, который запрашивает информацию на сервере. Этот запрос может попросить у сервера передачу какого-либо файла, страницы или скрипта, чтобы отобразить их в браузере.

HTTP(S) GET-флуд – метод DDoS-атаки прикладного уровня модели OSI, при котором злоумышленник посылает огромный поток запросов на сервер с целью переполнения его ресурсов. В результате сервер оказывается не способным отвечать не только на хакерские запросы, но и на запросы реальных клиентов.

- HTTP POST-флуд.

HTTP(S) POST-запрос – метод, при котором в тело запроса для последующей обработки на сервере помещаются данные. HTTP POST-запрос кодирует передаваемую информацию и помещает в форму, а затем отправляет этот кон-

тент на сервер. Данный метод используется для передачи больших объёмов информации.

HTTP(S) POST-флуд – это вид DDoS-атаки, при котором количество POST-запросов переполняют сервер таким образом, что сервер оказывается не в состоянии ответить на все запросы, что может привести к предельно высокому использованию системных ресурсов, а затем и к аварийной остановке сервера.

Описанные выше HTTP-запросы могут передаваться по защищённому протоколу HTTPS. В этом случае все данные, пересылаемые между клиентом (либо злоумышленником) и сервером, шифруются. Исходя из этого, «защищённость» идёт на пользу злоумышленникам, поскольку, чтобы выявить враждебный запрос, сервер должен сначала расшифровать его. Таким образом, приходится расшифровывать весь поток запросов, а их вовремя DDoS-атаки поступает чрезмерно много, что создаёт дополнительную нагрузку на сервер-жертву.

- SYN-флуд.

SYN-флуд (TCP/SYN) устанавливает полуоткрытые соединения с узлом. Когда жертва получает SYN-пакет через открытый порт, она должна отправить в ответ SYN-ACK пакет и установить соединение. После этого инициатор посылает получателю ответ с ACK-пакетом. Данный процесс условно называют рукопожатием, которое во время атаки SYN-флудом не может быть завершено, поскольку атакующий не отвечает на SYN-ACK сервера-жертвы. Такие соединения не будут закрыты до истечения тайм-аута, поскольку очередь на подключение переполняется, то новые клиенты не имеют возможности подключиться к серверу. Таким образом, SYN-флуд поражает сервер, занимая всю память таблицы соединений, которая обычно используется для хранения и обработки входящих пакетов. Все это может вызвать критическое падение производительности и, как следствие, отказ работы сервера.

- UDP-флуд.

UDP-флуд чаще всего применяется для широкополосных DDoS-атак по причине их бессеансовости.

- ICMP-флуд.

Протокол межсетевых управляющих сообщений (ICMP) используется для передачи сообщений об ошибках и не используется для передачи данных. ICMP-пакеты могут сопровождать TCP-пакеты при соединении с сервером. ICMP-флуд – метод DDoS атаки на сетевом уровне модели OSI, который использует ICMP-сообщения для перегрузки сетевого канала атакуемого.

- MAC-флуд.

MAC-флуд – довольно редкий вид атаки, при котором злоумышленник посылает множественные пустые Ethernet-фреймы с различными MAC-адресами. Сетевые свитчи рассматривают каждый MAC-адрес в отдельности и, как следствие, резервируют ресурсы под каждый из них. Когда вся память на свитче заполнена, он либо перестаёт отвечать, либо выключается. На некоторых типах роутеров подобная атака может стать причиной удаления таблиц маршрутизации, что нарушит работу целой сети.

Некоторые действия и оборудование для устранения атак:

- Брандмауэры с динамической проверкой пакетов.
- Динамические механизмы SYN прокси.
- Ограничение количества SYN-ов за секунду для каждого IP-адреса.
- Ограничение количества SYN-ов за секунду для каждого удаленного IP-адреса.
- Установка экранов ICMP флуда на брандмауэре.
- Установка экранов UDP флуда на брандмауэре.
- Ограничение скорости роутеров, примыкающих к брандмауэрам и сети.

В целом, общий подход и главная цель к защите от DDoS-атак заключается в следующем:

- обнаружение вторжения;
- определение источника вторжения;
- предотвращение вторжения.

III.2. Межсайтовый скриптинг (XSS)

Межсайтовый скриптинг или XSS (англ. Cross-Site Scripting) – тип атаки на web-системы, заключающийся во внедрении в выдаваемую web-системой страницу вредоносного кода (который будет выполнен на компьютере пользователя при открытии им этой страницы) и взаимодействии этого кода с web-сервером злоумышленника.

Для термина используют сокращение «XSS», чтобы не было путаницы с каскадными таблицами стилей, использующими сокращение «CSS».

Данная уязвимость появляется из-за недостаточной фильтрации данных, которые пользователь отправляет для отображения в web-ресурсе. В качестве примера можно привести любую гостевую книгу – программу, предназначенную для принятия данных от пользователя и последующей их вставки в web-страницу, причём гостевая книга никак не проверяет и не фильтрует вводимые данные, а просто их отображает.

Web-сайты хранят данные и постоянно отправляют информацию браузеру пользователя: атаки типа XSS происходят, когда ненадёжные источники отправляют пользователям вредоносное содержимое, пользуясь уязвимостями в web-сайтах с целью кражи персональных данных или нанесения ущерба компьютеру пользователя. Например, когда пользователь ищет какую-либо информацию онлайн, web-сайт отправляет браузеру данные в форме поисковой выдачи. При атаках межсайтового скриптинга передаваемая информация содержит вредоносный код и может способствовать злоумышленникам краже персональных данных пользователя. Благодаря тому, что почти каждый web-сайт хранит и отправляет данные браузерам, данный вид атаки на сегодняшний день представляет наиболее распространённую уязвимость программного обеспечения.

Обычно для организации этой атаки требуется взаимодействие с пользователем: либо его завлекают на заражённый сайт с помощью социальной инженерии, либо просто ожидают, пока пользователь сам посетит данный сайт. По этой причине разработчики часто не уделяют должного внимания XSS-уязвимости. Но если их не устранять, это может нести серьёзную угрозу без-

опасности, поскольку на странице или в HTTP-cookie могут быть весьма уязвимые данные (например, идентификатор сессии администратора или номера платёжных документов). Межсайтовый скриптинг предоставляет вредителю практически полный контроль над одним из важных представителей программного обеспечения в наши дни – браузером.

Виды XSS

К основным видам межсайтового скриптинга относят:

- постоянные (хранимые);
- непостоянные (отражённые);
- DOM-модели.

Постоянный XSS: вредоносный код хранится на сайте или сервере. Злоумышленником может быть введено специально созданное сообщение в гостевую книгу (это может быть комментарий, сообщение форума или профиль), которое сохраняется на сервере и загружается с него каждый раз, когда пользователи запрашивают отображение этой страницы. Или, к примеру, хакер получает доступ к данным сервера, например, через SQL-инъекцию, и внедряет в данные, которые выдаются пользователю, злонамеренный JavaScript код (с килогерами или с BeEF).

Непостоянный XSS: пользователю необходимо посетить специально сформированную ссылку. Например, на сайте имеется поиск, который вместе с результатами поиска пользователя отображает другие данные, например, «Вы ищете: [строка поиска]», при этом отображаемые данные не фильтруются верным образом. Поскольку такая страница отображается только для того, у кого есть ссылка на неё, то до тех пор, пока хакер не отправит ссылку другим пользователям сайта, атака не сработает. Вместо отправки ссылки пользователю, злоумышленники могут использовать размещение вредоносного скрипта на нейтральном сайте, который посещает жертва.

XSS в DOM-модели: источник проблемы находится на стороне клиента. Данный вид атаки является уязвимостью высокой степени опасности, посколь-

ку она может использоваться для изменения DOM-модели сайта, а это может повлечь похищение учётных данных администратора сайта и получение полного контроля над уязвимым приложением.

Некоторые действия и оборудование для устранения атак:

Обнаружить и устранить уязвимость типа XSS – это задача владельца web-сайта, поскольку именно на нем находится вредоносный код, который заражает ничего неподозревающих пользователей – посетителей данного сайта. Поскольку подобным уязвимостям в одинаковой степени подвержены как сайты с низкой, так и высокой репутацией, пользователям следует самим стараться обезопасить себя и свои данные. Помочь избежать межсайтового скриптинга могут следующие действия:

- установка качественного антивирусного программного обеспечения на свой компьютер;
- настройка автоматического обновления всех сторонних программ;
- загрузка инструмента для сканирования WEB-сайтов на предмет наличия в коде уязвимостей типа XSS.

III.3. SQL-инъекции

Внедрение SQL-кода (англ. SQL-injection) – это один из распространённых методов взлома web-приложений, работающих с базами данных, основанный на внедрении в запрос SQL-кода злоумышленника. В случае успеха атакующий может изменить логику выполнения SQL-запроса таким образом, как это ему нужно: выполнить произвольный запрос к базе данных (например, прочитать содержимое любых таблиц, добавить, изменить или удалить данные), получить возможность чтения и/или записи локальных файлов и выполнения каких-либо команд на атакуемом сервере. Атака типа внедрения SQL может быть возможна из-за некорректной обработки входных данных, используемых в SQL-запросах. Покажем на простом жизненном примере. Папа написал записку маме, чтобы она дала Васе 50 рублей, и положил её на стол. Сформируем это в

простой и незатейливый SQL-язык, получится: «Достань из кошелька 50 рублей и дай их Пете». Но отец написал записку неразборчиво, и на столе её увидел брат Пети – Вася. Вася, будучи хакером, добавил в записке текст «или Васе» и получился следующий запрос: «Достань из кошелька 50 рублей и дай их Пете или Васе». Мама, увидев записку, решила, что Пете она давала деньги вчера и дала 50 рублей Васе. Таким образом, мама не проверила данные – еле разобрала корявый почерк, а Вася извлёк выгоду.

Виды SQL-инъекций

Выделяют пять основных классов SQL-инъекций:

1. UNION query SQL injection. Классический вариант внедрения SQL-кода, когда в уязвимый параметр передаётся выражение, которое начинается с «UNION ALL SELECT». Данный способ работает, когда web-приложения напрямую возвращают результат вывода команды SELECT на страницу: с использованием цикла for или другим похожим способом, таким образом, что каждая запись выборки, полученной из базы данных, последовательно выводится на страницу.
2. Error-based SQL injection. В случае такой атаки сканер заменяет или дополняет уязвимый параметр синтаксически неверным выражением, после чего обрабатывает HTTP-ответ (заголовки и тело) в поиске ошибок DBMS, в которых могла бы содержаться заранее известная инъецированная последовательность символов и где-то «поблизости» вывод на интересующий нас подзапрос. Этот приём работает только тогда, когда web-приложение по каким-либо причинам, зачастую в целях отладки, раскрывает ошибки DBMS.
3. Stacked queries SQL injection. Сканер проверяет, поддерживает ли web-приложение последовательные запросы, и, если они выполняются, добавляет в уязвимый параметр HTTP-запроса точку с запятой «;» и затем внедряемый SQL-запрос. Этот метод используется

для внедрения SQL-команд, отличающихся от SELECT, например, для манипуляции данными (с помощью INSERT или DELETE). Следует отметить, что подобная атака в принципе может привести к возможности чтения или записи из файловой системы, а также выполнению команд в операционной системе.

4. Boolean-based blind SQL injection. Так называемая слепая инъекция – это логический метод SQL-инъекции, основанный на отправке запроса SQL в базу данных, который заставляет приложение возвращать другой результат в зависимости от того, возвращает ли запрос результат true или false. Эта атака, как правило, медленная (в особенности на большие базы данных), поскольку атакующему необходимо перечислять базу данных, символ за символом.
5. Time-based blind SQL injection. Это так называемая полностью слепая инъекция. Аналогично, как и в предыдущем случае, сканер работает с уязвимым параметром. Но в этом случае им добавляется подзапрос, который приводит к паузе работы DBMS на определённый отрезок времени (например, при помощи команд SLEEP() или BENCHMARK()). Используя эту особенность, сканер может посимвольно извлечь данные из базы данных, при этом сравнивая время ответа на запросы – оригинальный и с внедренным кодом. Здесь также используется алгоритм двоичного поиска. Помимо того, применяется специальный метод для верификации данных, с целью уменьшения вероятности неправильного извлечения символа из-за нестабильного соединения.

Некоторые действия и оборудование для устранения атак:

Разработчик прикладных программ, работающих с базами данных, должен знать о таких уязвимостях и принимать меры противодействия внедрению SQL.

Для проверки web-ресурса на наличие вредоносного программного обеспечения необходимо загрузить скрипт-антивирус на используемый хостинг и

запустить его. После этого нужно закрыть прореху, через которую злоумышленник попал на web-ресурс. Обновить программу для управления ресурсом, плагины, поменять пароли, по возможности обратиться к хостеру для получения лог-файла, по которому можно понять, каким образом был получен доступ. После закрытия канала проникновения и удаления бэкдоров, работа должна наладиться, но каждый случай является индивидуальным и уникальным, поэтому требует персонального подхода.

К наиболее надёжным способам предотвращения SQL-инъекций можно отнести использование параметризированных SQL-параметров. Основная задача заключается в том, что если изначально позицию параметров явно задать, то можно абсолютно безопасно передавать SQL-запросы базе данных, исключая возможность для параметров самим стать SQL-выражениями (в том числе злонамеренными). Проблема заключается в том, что существуют некоторые варианты для их обхода, а значит, к их использованию необходимо подходить с максимальным вниманием.

Самое основное – фильтрация входящих данных, например, в параметрах поиска, при выборе номера страницы и т.п.

Выделим основные моменты для защиты web-ресурса:

- максимально возможная фильтрация данных: атакующий не должен иметь прав вставить свою кавычку, удалить кавычку владельца ресурса;
- при сравнении использовать кавычки SELECT …WHERE name=’$name’;
- при использовании SQL-функции LIKE, необходимо обязательно фильтровать символы «%» и «_»;
- в случае, когда нужна более серьёзная защита, можно, как вариант, создать белый список вызова запросов, и web-ресурс будет отвечать только на них. Белый список (Whitelist) – один из лучших методов защиты от SQL-инъекций. Идея заключается в том, что в коде программы прописываются разрешённые значения параметров для пе-

редачи SQL-серверу, а это практически полностью исключает возможность взлома web-ресурса с помощью SQL-инъекций.

III.4. Перебор паролей

Как правило, большинство пользователей не любят сложные «правильные» пароли, а также использование разных паролей для разных ресурсов (например, платёжных систем, интернет-банков, учётным записям электронной почты и т.д.) – ведь запомнить один гораздо проще, чем несколько. Если создание простого пароля невозможно, то зачастую можно наблюдать следующую картину – пользователь хранит пароль, написанный на стикере, на экране монитора, а в лучшем случае, запиской, положенной под клавиатуру. С точки зрения безопасности последний вариант нельзя назвать самым худшим, поскольку пароль будет недоступен из сети, и все же делать так не следует. А вот в сети системы распознавания паролей могут быть подвержены довольно большому количеству проблем.

- Слабые пароли.

Заданный пользователем слабый или лёгкий пароль может быть абсолютно легко угадан злоумышленником, даже при отсутствии у него мощных эксплойтов. Атакующему достаточно «перебрать» стандартные, часто встречающиеся комбинации (например, [пустая строка], password, 12345, или хотя бы имя или дата рождения пользователя), и он получит доступ к ресурсу.

- Серийные пароли.

Иногда пользователю изначально даётся сложный пароль и просьба заменить его новым своим паролем. Однако, многие не утруждают себя этим, а ограничиваются небольшим изменением полученного пароля. Например, «TheSeptember03Password» заменяется на «TheOctober03Password» или «TheSeptember04Password». Логика в этом случае простая, поэтому подобное изменение пароля не несёт никакой сложности для злоумышленника.

- Передача паролей.

Проблема разглашения паролей является довольно актуальной. Это может произойти в результате обмана, подкупа или угроз. Использование одного и того же пароля в разных системах тоже можно отнести к этому виду проблемы, поскольку пароль, используемый в одной важной системе с хорошей защитой, может быть использован в другой, где допустимо хранение паролей в открытом виде или с применением ненадёжного шифрования. В целях избежать подобных атак можно сделать следующее. Если используется шифрование на уровне блоков, можно попытаться изменить длину пароля, чтобы для его хранения использовалось разное количество блоков. Рекомендуется проверить сценарии сбоев при установке: вызвать аварийное завершение и проверить файлы на диске на проверку хранения пароля в дампе.

- Пароли по умолчанию.

Пароли, присваиваемые по умолчанию, являются лучшими помощниками злоумышленников. Чаще всего данная проблема актуальна при работе с оборудованием, поскольку именно в этой области это наиболее оправдано.

- Неограниченный срок действия паролей.

Если не применять обязательную замену пароля через определённые временные промежутки, то пароль, подвергшийся взлому, останется действительным в течение неопределённо длительного времени.

- Атаки повторного воспроизведения.

Данный вид атаки основан на перехвате сетевого трафика между пользователем и серверным приложением, после чего злоумышленник снова отправляет перехваченные пакеты для получения того же уровня доступа, каким должен был обладать пользователь. В силу распространения беспроводных сетей перехват сетевого трафика уже не является сложной задачей, по причине того, что типичные беспроводные сети либо вообще не защищены, либо легко взламываются. Однако, проводные сети тоже могут быть подвержены атакам типа «незаконный посредник» (MITM, Man-In-The-Middle). Существует много видов атак повторного воспроизведения. В качестве простого примера можно приве-

сти попытку избежать пересылки текстовых паролей по сети в случае восстановления паролей по электронной почте. Пароль хешировался и передавался на сервер. В случае перехвата атакующий мог получить доступ к почте пользователя, не зная пароля. Ещё один вид атаки повторного воспроизведения после безопасной аутентификации связан с проведением дополнительной проверки со сравнением между клиентскими cookie и полученными от сервера. В качестве выявления данного вида атаки можно сохранить трафик аутентификации и попытаться войти в систему, отсылая тот же трафик. Если он инкапсулируется в SSL/TSL, то нужно создать посредника с другим сертификатом SSL и проверить, обнаружит ли клиентское приложение факт несовпадения сертификатов. Если это замечено не будет, то, скорее всего, трафик открыт для атак повторного воспроизведения. Основной мерой защиты от подобных атак является инкапсуляция попыток аутентификации по защищённому каналу, например, SSL/TSL или IPSec.

- Хранение паролей вместо хеш-кодов.

Пароли, которые хранятся на сервере, могут быть похищены различными способами: получены в результате атаки SQL-инъекции, украдены непорядочным работником, прочитаны из резервных копий баз данных и другими способами. Кроме нарушения безопасности системы пользователя, похищенный пароль позволяет злоумышленнику проникнуть в другие системы, посещаемые этим пользователем, а также открывает доступ к частной и личной информации.

- Атаки методом «грубой силы» против хеш-кодов паролей.

Довольно часто хеш-коды паролей создаются неправильно. Аспект первый: если для одного пароля всегда возвращается один и тот же хеш-код, хакер может легко определить, кто из клиентов использует одинаковые пароли, провести атаку методом «грубой силы» по словарю, когда для атаки используется файл, который содержит список слов, а программа проверяет каждое из слов, чтобы найти результат, а потом запустить более сложную атаку с помощью «радужных таблиц» (Rainbow tables), в этом виде атаки используются предва-

рительно вычисленные хеш-коды, поэтому она быстрее. Вторым аспектом является то, что проверка пароля может выполняться слишком быстро. Она не должна включать в себя большие вычислительные ресурсы, но временные затраты должны быть такими, чтобы атака стала неприемлемой для злоумышленника, пытающегося применить метод «грубой силы». Для тестирования производительности можно проверить, сколько попыток в секунду допускает система. Как правило, если она хорошо спроектирована, то система не разрешит сделать более нескольких сотен попыток в секунду. При проведении тестирования параллельно с анализом кода следует подсчитать количество операций хеширования, которые необходимы для создания контрольных данных пароля, оно должно быть не менее 1000, а желательно больше этого значения.

- Атаки реального времени.

При неэффективной и несовершенной реализации становится возможным быстрый перебор предположений, не замедляется процесс аутентификации при повторе неудачных попыток ввода и не предусматривается блокировка учётной записи при чересчур большом количестве попыток. Либо подобное проведение атаки основывается на постоянной или длительной блокировке входа при несложных именах пользователей или таких, которые легко угадать. В этом случае возникает ситуация отказа в обслуживании.

- Восстановление забытого пароля.

Собственно, хранить пароли вообще не рекомендуется, но если без этого не обойтись, то лучше всего будет создание нового, случайно сгенерированного пароля.

Некоторые действия и оборудование для устранения атак

В качестве рекомендаций можно выделить следующее:

- никогда не хранить пароли и работать только с хеш-кодами;
- для предотвращения фишинговых атак предусмотреть, чтобы страница входа была доступна только через SSl/TSL;

- web-приложение должно проверять выполнение требований к длине пароля и его сложности, а сам пользователь не должен устанавливать недостаточную длину и проверять, чтобы его имя не содержалось в пароле;
- при проверке истории паролей следует попробовать поменять в пароле все цифры и изменить другие символы: если алгоритм способен сформировать предыдущий пароль по введённому, значит злоумышленник сможет сделать то же самое, и от данного пароля следует отказаться;
- регулярно менять свои пароли, но не стоит это делать слишком часто – это чревато забыванием паролей, во всем нужна мера;
- не использовать пароли по умолчанию, в случае его необходимости, в частности, для работы с оборудованием, возможны следующие решения: либо перевести систему в режим ограниченной функциональности до установления другого пароля, либо запретить удалённый вход вплоть до изменения пароля по умолчанию;
- не хранить пароли в программном коде и во внутренней инфраструктуре в формате простого текста;
- использовать одноразовые пароли для подключения с ненадёжных систем.

III.5. Незащищённые ресурсы

Небезопасные прямые ссылки на объекты

Этот вид уязвимости выражается тогда, когда разработчик указывает на внутренний объект (файл, каталог или запись в базе данных, как параметр URL) прямую ссылку. Это делает возможным получение несанкционированного доступа к данным злоумышленнику с помощью различных манипуляций с данным параметром. Сюда можно отнести следующее: прямая ссылка на приватное изображение в закрытом каталоге или альбоме; открытый номер кошелька

в GET-запросе; AJAX запрос-ответ по userID, возвращающий все данные о пользователе в JSON, которые фильтруются на стороне клиента; редактирование или незакрытый просмотр своего профиля, например, пользователь с ID ***b*** может перейти на страницу юзера ID ***a*** и изменить его (рис. Ш.2.).

Рис Ш.2. Пример адреса профилей

В качестве борьбы с этой уязвимостью стоит отметить проверку, нужный ли пользователь находится на web-ресурсе, а также следует отнести к простой защите количество неверных попыток ввода пароля.

Небезопасная конфигурация

Небезопасная конфигурация – это неверно или недостаточно настроенный конфигурационный файл сервера. В качестве примера можно привести:

- открытый логин;
- незакрытое профилирование запросов;
- открытый файл конфигурации (например, main.inc вместо main.php);
- отсутствие аккаунта admin;
- открытый доступ к папке images (исправить ситуацию поможет .htaccess или пустой index.html).

Некоторые действия и оборудование для устранения атак.

В качестве профилактики и во избежание данного вида уязвимости можно перечислить следующие аспекты:

- актуальные версии используемого программного обеспечения;
- отсутствие загруженности модулями и плагинами;

- исключение возможности открытого показа фрагментов кода или различных операций, которые отвечают за какую-либо операцию файлов;
- установка различных паролей на различные сферы (панель хостера, SSH, MySQL);
- регулярная замена паролей;
- строгая архитектура приложения, в которой разработчик легко ориентируется.

Утечка чувствительных данных

В настоящее время информация является самым ценным активом человека, организации и страны в целом. Большое количество данных, в частности, конфиденциальных, хранится и обрабатывается практически во всех структурах – медицинских, финансовых, государственных. Количество подобных данных растёт со стремительной скоростью, а инструменты для их анализа и обработки быстро развиваются и меняются, что привлекает злоумышленников к обладанию такими данными. Здесь следует отметить два вида утечек – умышленные и неумышленные. Умышленные (преднамеренные) утечки характеризуются тем, что хакер, имея доступ к конфиденциальной информации, понимает противоправность своих действий и осознает возможные негативные последствия. Как правило, злоумышленники получают доступ к защищаемой информации ради материальной наживы. Неумышленная же утечка происходит вследствие халатности или неосведомлённости пользователей в вопросах защиты информации. Зачастую утечка информации происходит как следствие взлома web-ресурса, в результате которого были похищены данные.

Некоторые действия и оборудование для устранения атак.

- не хранить конфиденциальную информацию в открытом виде или в открытом доступе;
- использовать актуальные версии программного обеспечения;

- использовать надёжный алгоритм шифрования и периодически его обновлять.

Подводя итоги, следует отметить, что уязвимости в web-приложениях попрежнему остаются одним из наиболее распространённых недостатков обеспечения защиты информации. Проблема усугубляется ещё и тем, что при разработке web-приложений, зачастую не берутся в расчёт вопросы, которые связаны с защищённостью этих систем от внутренних и внешних угроз, либо этому процессу не уделяется достаточно внимания. Это может привести к ситуации, когда проблемы информационной безопасности обнаруживаются владельцем системы уже после завершения проекта. А устранить уязвимость в уже созданном web-приложении труднее и финансово более затратно, чем при его разработке и внедрении.

Одной из основных причин низкого состояния защищённости web-приложений, доступных в сети Интернет, является недооценка серьёзности риска реализации угроз информационной безопасности.

Общие рекомендации по защите WEB-ресурса

Надёжные пароли. Для генерации и хранения паролей желательно применять специальные менеджеры. Нельзя хранить пароли просто на компьютере, в браузерах и FTP-клиентах, нужно регулярно их менять.

Ограничение доступа. Список лиц, которые имеют доступ к учётной записи администратора WEB-ресурса, должен быть строго ограничен.

Безопасные протоколы. При подключении к сайту через файлообменные программы и прочие аналоги необходимо использовать только безопасные протоколы с шифрованием SFTP или SCP.

Сервер. Системные функции по возможности должны быть отключены, а WEB-ресурсы по максимуму изолированы. Учётную запись «по умолчанию» следует отключить, а в новой такой записи назначить минимум необходимых прав.

Программное обеспечение сервера. Требуется регулярно обновлять программное обеспечение, поскольку большая часть атак производится через уязвимости устаревшего программного обеспечения.

Файлы и каталоги. Файлы и каталоги должны быть доступны только для чтения. Необходимо запретить выполнение скриптов в каталогах загрузки.

Дополнения и компоненты. Все компоненты, которые не используются, необходимо отключить.

Антивирусные средства. Компьютер администратора WEB-ресурса должен быть максимально защищён, а сам ресурс должен периодически проверяться на наличие компьютерных вирусов.

Бэкапы. Необходимо регулярно создавать резервные копии сайта, а лучше сделать эту процедуру автоматической.

HTTPS. Использовать безопасный протокол шифрования данных при оплате или передаче конфиденциальных (и не только) данных.

CMS. Выбирать надёжные системы управления сайтом, которые имеют встроенные системы защиты, а также регулярно их обновлять.

Аудит безопасности. Для предупреждения уязвимостей можно проводить аудит с помощью программ-сканеров, либо привлекать специалистов.

Аудит безопасности – это упреждающая мера, которая позволяет выявить уязвимости, оценить защищённость WEB-ресурса и возможные сценарии атак, а после проверки сформировать рекомендации по защите ресурса.

Комплексный аудит безопасности включает в себя:

- атаку подбором паролей;
- внедрение XML-сущностей;
- поиск компонентов с известными уязвимостями;
- проверку на удаленное выполнение произвольного кода;
- поиск уязвимостей серверных компонентов и в WEB-окружении сервера;
- проверку на наличие «инъекций» кода;
- попытки обхода системы аутентификации;

- поиск XSS/ CSRF-уязвимостей;
- попытки перехвата привилегированных аккаунтов или их сессий;
- проверку на возможность файловых инъекций Remote File Inclusion/ Local File Inclusion;
- проверку на открытые редиректы и перенаправления на другие сайты;
- сканирование директорий с помощью перебора и взлома;
- анализ всех форм на WEB-ресурсе: регистрации, авторизации, поиска и т.д.;
- проверку на возможность открытого получения конфиденциальной информации;
- проверку на атаки класса Race Condition – ошибки проектирования многопоточных систем и приложений.

Для подобной проверки используются сканеры уязвимостей, так называемые специальные программы, проверяющие web-ресурс и зондирующие типичные уязвимые места, тем самым анализируя общую защищённость web-ресурса.

Сканеры бывают 3-х типов:

1. Сетевые. Проверяют ресурс дистанционно, подключаясь через сетевые сервисы. Этот вид сканеров является самым популярным.
2. Пассивные. В качестве источника данных используют сетевой трафик, но в отличие от сетевых, минимизируют влияние сканера на уязвимости.
3. Локальные. Устанавливаются непосредственно на проверяемом узле, благодаря этому аспекту обеспечивая высокую достоверность. Ищут уязвимости, сравнивая атрибуты файлов.

Обеспечение безопасности сайта – постоянный процесс, который не ограничивается проверками. По результатам сканирования ресурс необходимо модернизировать, дорабатывать, закрывать бреши, а какие-то вопросы решать на стороне сервера. Аудит безопасности позволяет определить ряд мер и работ,

которые надо выполнить. Это могут быть доработки программной части, услуги инженеров, которые решат технические вопросы, и набор организационных мер.

IV. Особенности безопасного конфигурирования серверных приложений

IV.1. SSH-доступ

В этом параграфе обозначены ключевые этапы обеспечения удалённого доступа через SSH-протокол.

SSH (англ. Secure SHell – защищённая оболочка) – сетевой протокол прикладного уровня, предназначенный для безопасного удалённого доступа к UNIX-системам. Данный протокол эффективен тем, что шифрует всю передаваемую информацию по сети. Главным образом протокол используется для удалённого управления данными пользователя на сервере, запуска служебных команд, работы в консольном режиме с базами данных, а также для создания туннелей – защищённого логического соединение между двумя узлами сети. Следует заметить, что существуют реализации SSH-протокола для самых различных аппаратных платформ и операционных систем.

Основные функции SSH

Протокол SSH возник в 1995 году как попытка обезопасить открытые незащищённые соединения. Наиболее важными функциями SSH на сегодняшний момент являются:

- Безопасные команды доступа к хосту. SSH даёт возможность выполнять безопасные команды доступа к хосту, такие как ssh (удалённая оболочка), slogin (удалённый вход в систему), scp (удалённое копирование).
- Туннелирование каналов внутри установленного соединения.
- X11 Forwarding. SSH предоставляет встроенный механизм для выполнения удалённых клиентов X Window System (графическая среда UNIX-подобных ОС).
- Сжатие передаваемых данных.

Безопасность использования SSH-протокола, обеспечивается следующими решениями:

- Шифрование соединения. В SSH используется ассиметричное шифрование (соответственно используется пара из закрытого и открытого ключа), можно выбрать алгоритмы для шифрования передаваемых данных. Опции команды SSH указаны в таблице IV.1. Шифрованное соединение не позволяет просто перехватить и использовать трафик.
- Аутентификация сервера выполняется при любом соединении. Это не позволяет выполнить подмену сервера или подмену трафика.
- Аутентификация клиента может выполняться одним из нескольких доступных способов. Это, с одной стороны, может повысить надёжность аутентификации и систему более гибкой и упрощает её использование.
- Проверка целостности пакетов позволяет отследить любые незаконные изменения в трафике соединения. При обнаружении таких изменений, соединение немедленно разрывается.
- Временные параметры аутентификации не позволяют воспользоваться данными соединения в том, случае, если спустя некоторое время после перехвата оно все-таки было расшифровано.

Рассмотрим установку и настройку SSH на примере работы в ОС Linux

Для работы по протоколу SSH необходимы: SSH-сервер и SSH-клиент. Сервер прослушивает соединения от клиентских машин и при установлении связи производит аутентификацию, после чего начинает обслуживание клиента. Клиент используется для входа на удалённую машину и выполнения команд.

Если необходимо подключение к SSH-серверу, установленному на Linux-машине, а на рабочая станция работает под управлением ОС Windows, то можно использовать бесплатный SSH/telnet – клиент PuTTY. Есть версия PuTTY и для разновидностей ОС Linux.

Таблица IV.1. Опции команды SSH

-c blowfish|3des|des	Служит для выбора алгоритма шифрования, при условии, что используется первая версия протокола SSH (blowfish, des или 3des – это название алгоритмов шифрования).
-c <шифр>	Задает список шифров, разделенных запятыми в порядке предпочтения. Опция используется для второй версии.
-f	Переводит ssh в фоновый режим после аутентификации пользователя.
-l <имя_пользователя>	Указывает пользователя, от имени которого нужно зарегистрироваться на удаленном компьютере. Опцию использовать необязательно, поскольку удаленный компьютер и так запросит имя пользователя и пароль.
-p порт	Определяет порт SSH-сервера (по умолчанию используется порт 22).
-q	Устанавливает Тихий режим. Будут отображаться только сообщения о фатальных ошибках. Все прочие предупреждающие сообщения в стандартный выходной поток выводиться не будут.
-x	Отключает перенаправление X11.
-X	Задействует перенаправление X11.
-1	Разрешает использовать первую версию протокола SSH.
-2	Разрешает использовать только вторую версию протокола SSH. (Вторая версия протокола более безопасна).

В Linux-системах используется метапакет openSSH, в котором реализованы две версии протокола SSH: SSH1 и SSH2 в виде открытого и свободно рас-

пространяемого ПО. Базовый метапакет SSH включает в себя сервер SSH (sshd) в качестве программы-демона, а также несколько утилит:

- ssh – удаленная регистрация и выполнение команд;
- scp – передача файлов;
- ssh-keygen – для генерации пар SSH-ключей.

Для установки всех требуемых утилит ssh нужно выполнить следующую команду (пример для ОС Ubuntu Linux):

$ sudo apt-get install ssh

Для соединения сервера и клиента необходимо создать пары ключей: открытый и закрытый и обменяться открытыми ключами. Обычно используется также и пароль.

Для подключения к удаленному узлу нужно выполнить следующую команду (пример для ОС Ubuntu Linux):

$ ssh [опции] <адрес_удаленного_узла>

В качестве адреса можно указать как IP-адрес, так и доменное имя узла.

При установке SSH-сервер автоматически прописывается в автозагрузку. Управлять его запуском, остановкой или перезапуском можно с помощью команды (пример для ОС Ubuntu Linux):

$sudo service ssh stop|start|restart

Основной файл конфигурации SSH-сервера – это файл /etc/ssh/sshd_config, доступный для чтения или редактирования только root. После каждого изменения этого файла необходимо перезапустить ssh-сервер для применения изменений.

Неправильно настроенный SSH-сервер – огромная уязвимость в безопасности системы, т.к. у возможного злоумышленника есть возможность получить практически неограниченный доступ к системе.

Рассмотрим и прокомментируем основные директивы конфигурационного файла.

Таблица IV.2. Основные параметры настройки SSH-сервера

Указывается порт, на котором будет работать SSH-сервер	
Port 1022 *Port 23*	Если директива не указана (закомментирована), то по умолчанию используется порт 22. Можно указывать несколько портов. Рекомендуется использовать нестандартный порт, т.к. стандартный часто сканируется ботами злоумышленников на предмет потенциальных уязвимостей.
Локальный адрес, который должен прослушиваться SSH-сервером	
ListenAddress 0.0.0.0	Адрес можно записывать в формате: *ListenAddress host|IPv4_addr|IPv6_addr* *ListenAddress host|IPv4_addr:port* *ListenAddress [host|IPv6_addr]:port* Если порт не задан, *sshd* будет слушать на этом адресе и на порту, указанному в опции *Port*. Можно указывать несколько адресов.
Указывает, какое семейство IP адресов должно быть использовано	
AddressFamily inet	Варианты: *any* – любые inet – только *IPv4* *inet6* – только *IPv6* По умолчанию – *any*.
Директива, которая позволяет выбрать версию протокола	
Protocol 2	Рекомендуется выбрать вторую версию протокола.
Ключевые файлы для протокола SSH2	
HostKey */etc/ssh/ssh_host_rsa_key* *HostKey*	Указывает файл, содержащий закрытый хост-ключ используемый SSH2.

/etc/ssh/ssh_host_dsa_key	
Указывает алгоритмы шифрования, разрешенные для SSH2	
Ciphers aes128-cbc, 3des-cbc, …	Поддерживаемые алгоритмы *3des-cbc, aes128-cbc, aes192-cbc, aes256-cbc, aes128-ctr, aes192-ctr, aes256-ctr, arcfour128* и др.
Разрешение/запрет использования публичный ключей	
PubkeyAuthentication yes	Наиболее предпочтительным способом авторизации является аутентификация на основе SSH2 *RSA-ключей*. При таком способе пользователь генерирует на своей стороне пару ключей, из которой один ключ является секретным, а другой публичным. Публичный ключ копируется на сервер и служит для проверки идентичности пользователя. По умолчанию – *yes*.
Указывает файл, в котором содержатся публичные ключи, используемые для аутентификации пользователя	
AuthorizedKeysFile %h/.ssh/authorized_key или *AuthorizedKeysFile /etc/ssh/authorized_keys*	Пример для файла-ключа в домашней папке пользователя. Пример для файла-ключа в папке /etc/ssh/
Указание серверу разделять привилегии пользователя (Privilege Separation is turned on for security)	

UsePrivilegeSeparation yes	Указывает, должен ли сервер разделять привилегии. Если «должен» («*yes*»), то сначала будет создан непривилегированный дочерний процесс для входящего сетевого трафика. После успешной авторизации будет создан другой процесс с привилегиями вошедшего пользователя. Основная цель разделения привилегий – предотвращение превышения прав доступа. Значение по умолчанию – *yes*.
Разрешается/запрещается удаленный доступ пользователя *root*	
PermitRootLogin yes	Указывает, возможен ли ssh-вход под пользователем *root*. Может принимать значения: *yes* – *root* может зайти (обычная схема аутентификации); *without-password* – *root* может зайти, и парольная аутентификация для него будет отключена; *forced-commands-only* – *root* сможет зайти, пользуясь аутентификацией на основе публичного ключа, и только если передаст необходимую к исполнению команду. Это удобно для выполнения резервного копирования, даже в том случае, когда вход не через SSH для *root* запрещен. Все остальные методы аутентификации и команды для пользователя *root* будут заблокированы; *no* – *root* не может использовать SSH для в хода в систему. Значение по умолчанию – *yes*.
Проверять/не проверять режимы доступа и владения пользовательских файлов	
StrictModes yes	Указывает должен ли сервер проверить режимы

	доступа и владения пользовательских файлов перед тем, как дать пользователю войти. Обычно это объясняется тем, что новички часто делают свои файлы доступными для записи всем. По умолчанию – *yes*.
Списки пользователей (групп) которым можно/нельзя пользоваться сервером	
DenyUsers login1	Список пользователей, которым нельзя пользоваться сервером. По умолчанию – не указан (не запрещен никто).
AllowUsers login2	Список пользователей, которым можно пользоваться сервером. По умолчанию – не указан (разрешено всем). Ели указан хотя бы один пользователь, то к серверу есть доступ только для пользователей из списка.
DenyGroups group1	Список групп, которым нельзя пользоваться сервером По умолчанию – не указано.
AllowGroups group2	Список групп, которым можно пользоваться сервером. По умолчанию - не указан (разрешено всем). Если указана хотя бы одна группа, то только пользователям этой группы разрешен доступ к серверу.
Разрешение/запрет парольной аутентификации	
	Разрешенная по умолчанию парольная аутентификация является практически самым примитивным способом авторизации на сервере. С одной

PasswordAuthentication no	стороны это упрощает конфигурацию и подключение новых пользователей (пользователю достаточно знать свой системный логин/пароль), с другой стороны пароль всегда можно подобрать, а пользователи часто пренебрегают созданием сложных и длинных паролей. Настоятельно не рекомендуется использовать парольную аутентификацию
Запрет/разрешение аутентификации без пароля	
PermitEmptyPasswords no	В случае разрешенной аутентификации с помощью пароля указывает, возможен ли вход с пустым паролем. По умолчанию – *no*.
Указывает время авторизации	
LoginGraceTime 120	Время, по истечении которого сервер отключает пользователя, если тот не смог успешно авторизоваться. Значение *0* – разрешает пользователю авторизовываться бесконечно. По умолчанию – *120* (секунд).
Указывает максимальное число попыток аутентификации	
MaxAuthTries 6	Разрешенное для одного соединения максимальное число попыток аутентификации. Как только число неудачных попыток превысит половину заданного значения, все последующие попытки будут заноситься в журнал. Значение по умолчанию – *6*.
Указывает максимальное число одновременных подключений для сервера	
MaxSessions 10	По умолчанию – *10*.

Если в конфигурационном файле SSH-сервера включить «аутентификация по ключу». В таблице IV.2. это параметр PubkeyAuthentication. Тогда для аутентификации будет использоваться асимметричное шифрование, следовательно, необходимо сгенерировать пару ключей: секретный и публичный. Публичный ключ следует сохранить на серверном узле, а секретный поместить на клиентском узле. При попытке соединения клиента и сервера происходит сопоставление этой пары ключей (проверка на соответствие).

Следующая команда позволяет генерировать пару ключей (пример для ОС Ubuntu Linux), где параметр dsa или rsa определяет способ шифрования из таблицы IV.3.:

```
$ ssh-keygen - t dsa //или
$ ssh-keygen -t rsa //или
$ ssh-keygen -t ed25519
```

После генерации ключей образуются два файла: ~/.ssh/id_rsa – секретный ключ, и ~/.ssh/id_rsa.pub – публичный ключ, который следует поместить на удалённый SSH-сервер. При генерации пары ключей можно задать пароль на использование ключа, который при необходимости можно заменить командой:

```
$ ssh-keygen -f ~/.ssh/id_rsa -p
```

Публичный ключ можно скопировать на SSH-сервер следующей командой (пример для ОС Ubuntu Linux):

```
$ ssh-copy-id username@hostname // или
$ ssh-copy-id –p номер_порта username@hostname // если номер порта не 22
```

Следует обратить внимание, что для выполнения операции по копированию публичного ключа на удалённый SSH-сервер должна работать аутентификация по паролю (PasswordAuthentication). В дальнейшем её можно будет отключить и включить аутентификацию по ключу (PubkeyAuthentication). Если при генерации пары ключей, был указан пароль, то в процессе аутентификации на SSH-сервере запрашивается не пароль удалённого пользователя, а пароль на этот ключ, что делает SSH-сессию ещё более безопасной.

Таблица IV.3. Типы ключей. Способы шифрования

RSA (Rivest-Shamir-Adleman, алгоритм с открытым ключом) является одним из первых криптосистем с открытым ключом и широко используется для безопасной передачи данных. Безопасность основана на целочисленной факторизации. По сравнению с DSA RSA быстрее проверяет подпись, но медленнее для генерации. Эти ключи используются чаще других.
DSA (Digital Signature Algorithm, алгоритм цифровой подписи) является федеральным стандартом обработки информации для цифровых подписей. Безопасность зависит от дискретной логарифмической проблемы. По сравнению с RSA DSA быстрее генерирует подпись, но медленнее для проверки.
ECDSA (Elliptic Curve Digital Signature Algorithm, алгоритм цифровой подписи кривых) является реализацией эллиптической кривой DSA. Криптография с эллиптической кривой может обеспечить относительно тот же уровень безопасности, что и RSA, с меньшим ключом.
EdDSA (Edwards-Curve Digital Signature Algorithm, алгоритм цифровой подписи Эдвардса) – это схема цифровой подписи с использованием вариант подпись Schnorr на основе Кривые Twisted Edwards. Создание подписи является детерминированным в EdDSA, и его безопасность основана на неразрешимости определенных задач с дискретным логарифмом, поэтому это безопаснее, чем DSA и ECDSA.
Ed25519 (Edwards-Curve, алгоритм цифровой подписи эллиптической кривой) это схема подписи EdDSA, но с помощью SHA-512/256 и Curve25519 для шифрования используется защищённая эллиптическая кривая, которая обеспечивает лучшую безопасность, чем DSA и ECDSA.

SSH- туннелирование

В этом параграфе обозначены основные подходы решения проблемы быстрого и безопасного удалённого доступа к программам серверных приложений средствами SSH-туннелирования.

Обозначим проблему. Требуется с домашней рабочей станции или с узла публичной сети Internet быстро и безопасно получить доступ к программным модулям (ресурсы серверов), которые расположены на узлах удалённой сети. Удалённая сеть может быть корпоративной, и защищена сетевыми фильтрами и межсетевыми экранами.

Для защищённого удалённого доступа к конфигурационным файлам и программам серверных приложений можно использовать туннелированные, которые позволяют создавать защищённое логическое соединение между двумя узлами сети. Суть туннелирования состоит в том, чтобы инкапсулировать передаваемые пакеты протокола, вместе со служебными полями, в новый пакет («конверт») для обеспечения конфиденциальности и целостности всех передаваемых данных. Туннелирование используют для создания защищённого логического соединение между двумя узлами сети.

IV.2. FTP-доступ

Установка и настройка FTP-сервера для ОС Windows

Основные понятия раздела

FTP-протокол – это один из старых, но востребованных разработчиками и пользователями протоколов. FTP (File Transfer Protocol) разработан для передачи файлов. Этот протокол служит для обмена файлами между компьютерами через интернет или локальную сеть. Конечно, файлы можно передавать по электронной почте или через общедоступный (публичный) файлообменник. Но, если необходимо переслать (передать по сети) коллекции фотографий, фильмов достаточного объёма, то воспользоваться электронной почтой будет не так уж просто. Публичный файлообменник, во-первых, не является безопасным средством для передачи данных, а во-вторых, может создавать некоторые неудоб-

ства для своих пользователей, например, ограничение скорости, просмотр рекламы, ожидание таймера и т.д. Поэтому в данном случае, одним из приемлемых решений является установка и настройка собственного FTP-сервера.

FTP-сервер – это специальная программа, которая предоставляет пользователям возможность подключения и выполнения различных действий с хранящимися на сервере файлами и папками: скачивать, загружать, создавать, редактировать, переименовывать, удалять, назначать права доступа. FTP-сервер обладает определёнными особенностями:

1. Необходима аутентификация пользователей (хотя есть и анонимные FTP-сервера, которые не требует предоставления персональных данных: логина и пароля).
2. Для выполнения манипуляций создаётся специальная сессия, все действия выполняются в рамках этой текущей сессии.
3. Для каждого подключения клиента к серверу создаётся отдельный канал.
4. Данные можно передавать текстом или бинарным кодом.
5. Также можно выполнять разные манипуляции с файловой системой на сервере (копирование, удаление, поиск, создание файлов и каталогов).

FTP-клиент – это программа, позволяющая подключиться к удалённому серверу по протоколу FTP, выполнять на сервере действия с файлами. В качестве клиента может использовать браузер, либо специальную программу, например, FileZilla. Как правило, у специализированных программ, которые выступают в роли клиента, возможностей для работы с файловой системой FTP-сервера больше, чем у обычного браузера.

Установить FTP-сервер (FTP-службу) можно практически на любую операционную систему (Windows, Linux, MAC, Android и другие ОС). Для установки FTP-сервера можно воспользоваться соответствующим программным обеспечение или воспользоваться стандартными средствами самой операционной системы (например, в Windows XP, 10, 8, Vista есть такая возможность). За

время развития данного протокола появилось множество платных и бесплатных приложений, через которые можно его установить и настроить.

Популярность этой службы обусловлена многими фактами, в том числе и простотой её установки и настройки. Клиентам для входа на FTP-сервер достаточно любого браузера или для пользователей операционной системы Windows обычного проводника.

В рамках данного раздела рассмотрим безопасную установку и настройку FTP-сервера для операционных систем Windows и Ubuntu Linux.

Следует отметить, что если использовать FTP-протокол, то в этом случае данные между клиентом и сервером передаются без шифрования трафика. На практике совместно с протоколом FTP используются криптографические протоколы, например SSL/TLS, которые позволяют шифровать передаваемые данные.

Для создания и настройки FTP-сервера в операционной системе Windows можно воспользоваться встроенной функцией Internet Information Services (IIS). Данная функция позволяет обеспечить доступ из сети к указанной папке.

Установка и настройка FTP-сервера для Windows

Для установки сервера необходимо выполнить следующие шаги:

1. Активировать службу IIS.
2. Установить FTP-сайт (интерфейс для сервера).
3. Прописать правила для Windows Firewall (брандмауер).
4. Выбрать метод проверки подлинности пользователей и правила для аутентификации (а также следует позаботиться о создании самих пользователей в системе и/или их групп).

Активация службы IIS

Так как некоторые стандартные возможности ОС Windows отключены по умолчанию, следует активировать необходимые для установки сервера службы IIS, а именно для компонента «FTP-сервер»: «Расширяемость FTP» и «Служба FTP», для компонента «Средства управления web-сайтом»: «Консоль управления IIS».

Для этого следует выполнить следующие шаги:

- открыть «Панель управления», перейти в раздел «Программы и компоненты»;
- выбрать «Включение или отключение компонентов Windows» (рис. IV.1.);
- включить требуемые компоненты службы.

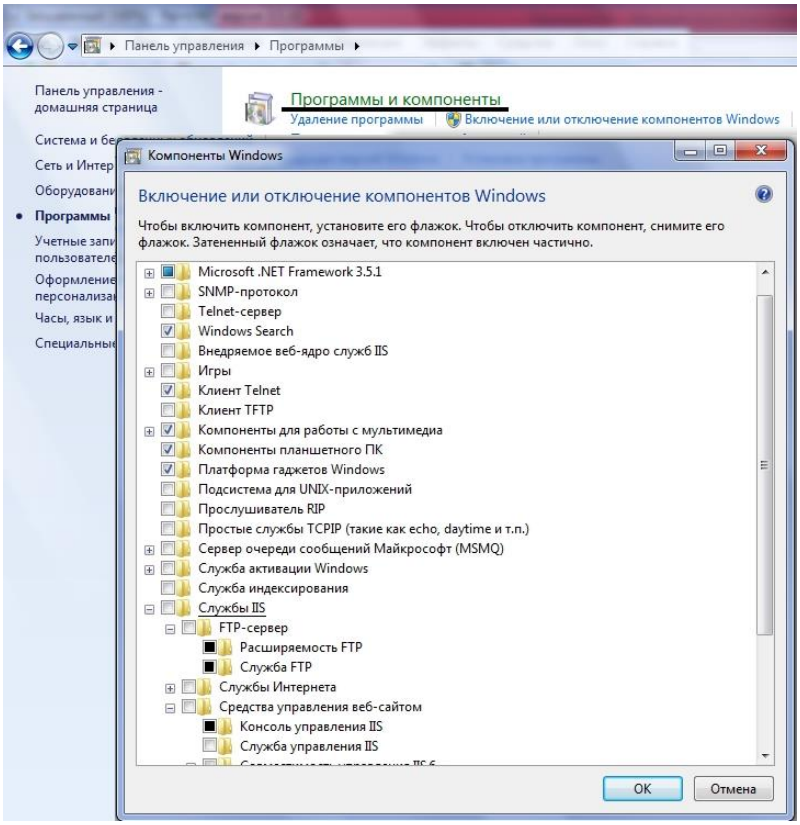


Рис. IV.1. Включение компонент IIS для установки FTP-сервера

Настройка FTP-сервера

Следующим шагом следует активировать сам сервер, отметить основные ошибки настройки, которые могут повлиять на его безопасную работу. Для активации и настройки FTP-сервера необходимо выполнит следующие шаги:

Через «Панель инструментов» перейти в раздел «Администрирование», а затем в подраздел «Диспетчер служб IIS».

В окне элемента «сайты» на панели «Подключения» в окне «Диспетчер служб IIS» следует выбрать действие «Добавить FTP-сайт» (рис. IV.2.). Выбранной командой запускается мастер установки FTP-сервера.

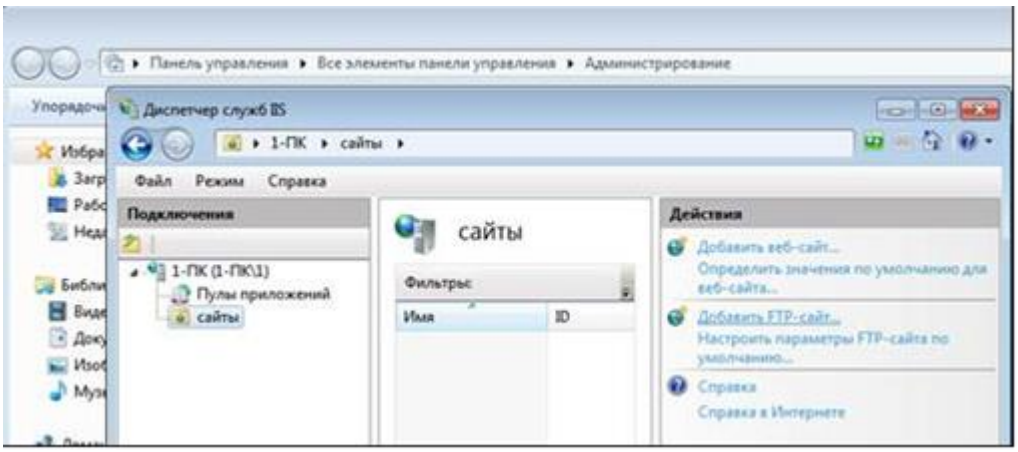


Рис. IV.2. Активация FTP-сайта

В окне «Добавить FTP-сайт» требуется указать имя нового сервера, и физический путь к корневому каталогу с его данными (рис. IV.3.). А затем перейти к следующему этапу настройки.

FTP-сайт выступает в роля интерфейса для работы пользователей с FTP-сервером.

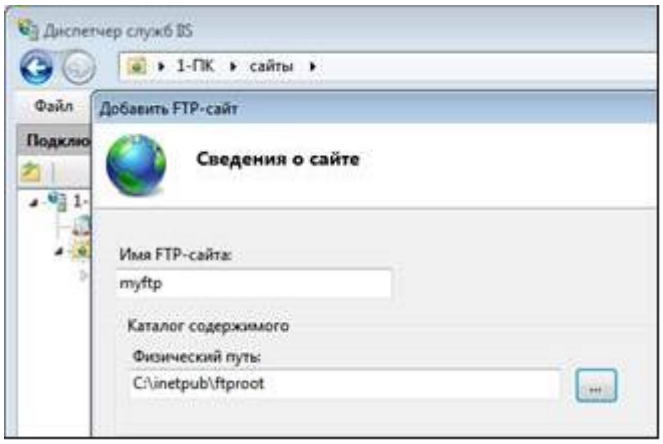


Рис. IV.3. Этап настройки FTP-сервера

Далее следует выбрать привязку к IP-адресу сервера и порт. По умолчанию осуществляется привязка ко всем свободным адресам, поле порт имеет значение по умолчанию 21.

Если другие FTP-сайты уже установлены на данном хосте, то с помощью определения виртуального узла можно сделать новый сайт для FTP-сервера уникальным. Для этого нужно установить опцию «Разрешить имена виртуальных узлов» и заполнить поле «Виртуальный узел».

Опция «Запускать FTP-сайт автоматически», определяет поведение сайта при перезагрузке хоста. По умолчанию опция установлена. Далее следует определить использование протокола SSL:

«Без SSL» – не использовать протокол защиты (не рекомендуется).

«Разрешить» – использовать SSL в случае, если FTP-клиент настроен на данный режим работы, в противном случае, не использовать данный протокол защиты.

«Требовать» – использовать SSL в любом случае. Если клиент не настроен на использование SSL протокола – отклонить соединение.

Следует выбрать необходимые настройки и перейти к следующему этапу.

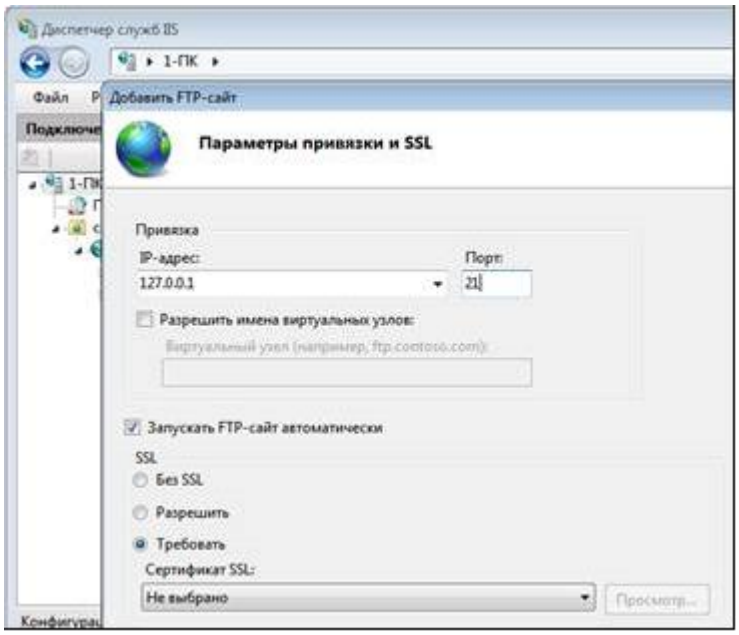


Рис. IV.4. Настройка FTP-сервера

На следующем шаге следует задать тип авторизации (рис. IV.5.). В пункте «Проверка подлинности» можно разрешить вход для обычных или анонимных пользователей. Если нужно подключать к серверу только определённых пользователей или пользователей определённых групп, тогда следует выбрать «вход для обычных пользователей». К настройке правил для авторизации следует вернуться позже, после выбора протокола авторизации, создания специальных групп и/или пользователей для работы с сервером.

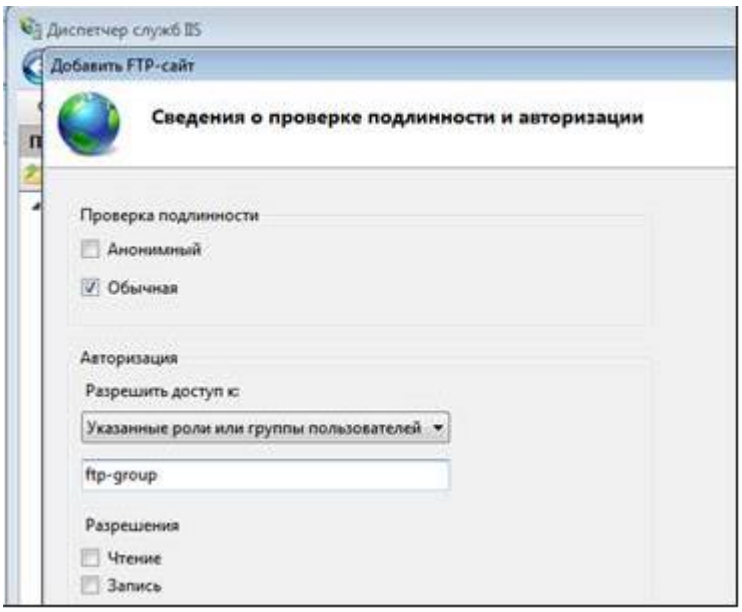


Рис. IV.5. Настройка FTP-сервера

После завершения работы мастера установки в окне «диспетчер служб IIS» появилась ссылка на интерфейс для нового FTP-сервера, но данная служба ещё не готова к работе. Для того, чтобы сервер начал функционировать, следует настроить Windows Firewall (брандмауер), определить протокол аутентификации и при необходимости создать новые группы и пользователей, а также следует настроить права доступа (разрешения NTFS) для каталогов FTP-сайта.

Настройка Windows Firewall (брандмауер)

По протоколу FTP возможно соединение в двух режимах – активном и пассивном. Для работы в пассивном режиме достаточно открыть порт – по умолчанию 21. В пассивном режиме по 21 порту происходит входящее соединение, затем клиент и сервер используют механизм трёхстороннего рукопожатия, договариваются, по какому порту будет происходить обслуживание и переключаются на него. Открытие целого диапазона портов, повлияет на безопасность сервера, поэтому для безопасной работы сервера в пассивном режиме нужно настроить входящее правило.

На следующем шаге сформулируем входящее правило, для этого следует перейти в настройку «Брандмауэр Windows в режиме повышенной безопасности», а затем следует создать правило для входящих подключений (рис. IV.6.).

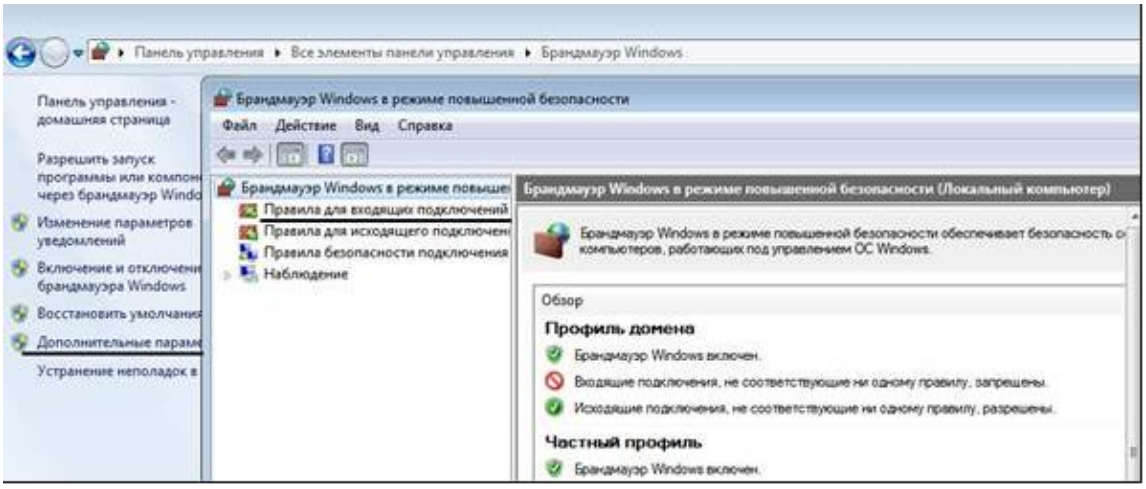


Рис. IV.6. Создание нового правила брандмауэра

С помощью мастера создаём новое правило с типом «предопределенные» с названием «FTP-сервер» (рис. IV.7.).

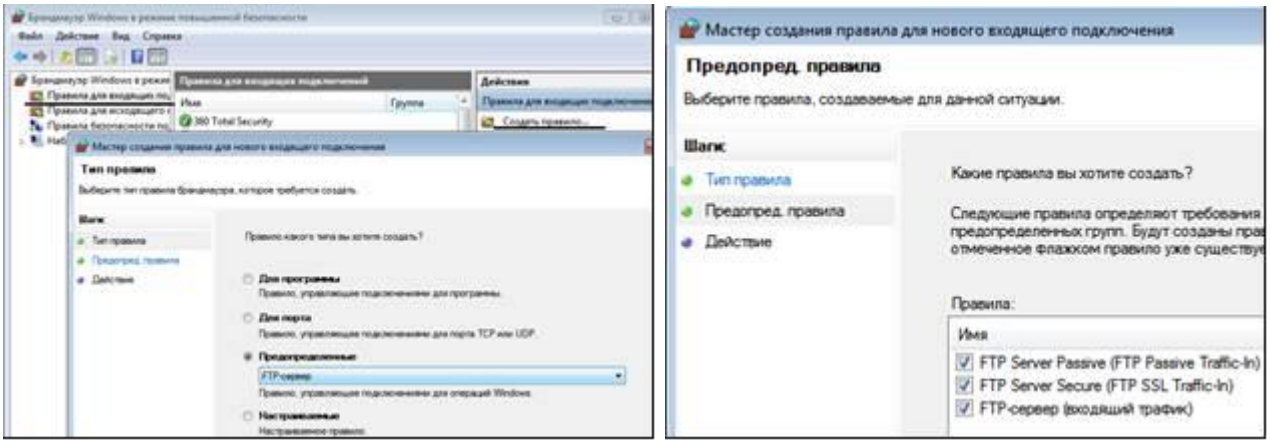


Рис. IV.7. Создание правила для брандмауэра

Затем на следующем шаге выбираем следующие предопределённые правила:

- FTP ServerPassive (FTP PassiveTraffic-In) – правило используется для поддержки входящих соединений в пассивном режиме.
- FTP ServerSecure (FTP SSL Traffic-In) – правило используется для поддержки входящих соединений по протоколу SSL.
- FTP-сервер (входящий траффик) – правило используется для поддержки входящих соединений по протоколу FTP.

Все три правила обеспечивают работу FTP-сервера. Второе правило используется для защиты траффика по протоколу SSL

На завершающем шаге мастера следует выбрать действие для правила. Доступны следующие значения:

- Разрешить подключение – включая подключения, защищённые IPSec, так и подключения без защиты.
- Разрешить безопасное подключение – включая только подключения с проверкой подлинности с помощью IPSec. Подключения возможны с клиентских операционных систем, начиная с Windows Vista.
- Блокировать подключения.

После выбора действия, новое правило для брандмауэра будет создано. Необходимо заметить, что новые правила вступят в силу только при перезагрузке хоста, на котором установлен сервер.

На следующем шаге настройки FTP-сервера необходимо настроить «Проверку подлинности FTP» (выбрать протокол аутентификации), для этого следует перейти к настройке возможностей созданного FTP-сайта (рис. IV.8.) (открыть окно «Диспетчер служб IIS», указать FTP-сайт, выбрать компонент «Проверка подлинности FTP»).

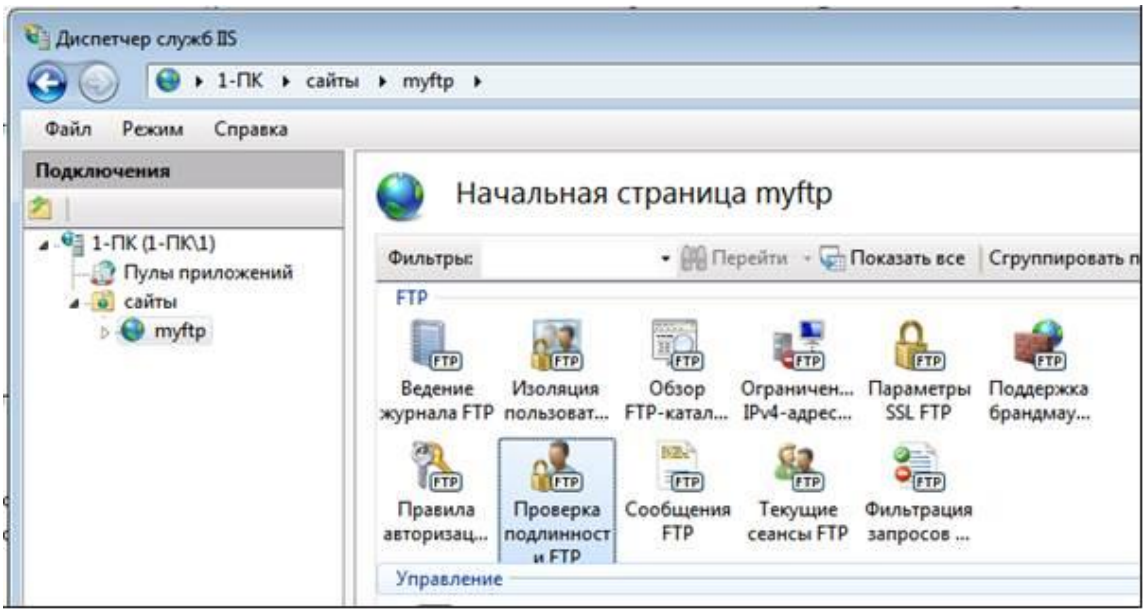


Рис. IV.8. Компоненты для настройки FTP-сайта

Страница «Проверка подлинности FTP» позволяет настроить методы проверки подлинности, которые могут быть использованы клиентами FTP для получения доступа к содержимому. По умолчанию все методы проверки подлинности отключены; необходимо включить какой-либо метод проверки подлинности, чтобы обеспечить доступ пользователям FTP к вашим данным. Предусмотрено два типа методов проверки подлинности: встроенные и настраиваемые.

Встроенные методы проверки подлинности входят в состав FTP-сервера. Эти методы проверки подлинности можно включить или отключить, но их нельзя удалить с FTP-сервера.

Настраиваемые методы проверки подлинности реализованы в виде устанавливаемых дополнительных компонентов. Эти методы проверки подлинности можно включить или отключить, а также их можно добавить на FTP-сервер или удалить с него.

К встроенным методам относятся:

1. Анонимная проверка подлинности – это метод проверки подлинности, который обеспечивает доступ любому пользователю к любому общему содержимому путём предоставления анонимного имени пользователя и пароля. По умолчанию анонимная проверка подлинности отключена. Анонимную проверку подлинности рекомендуется использовать, если необходимо, чтобы все клиенты, которые посещают FTP-сайт, имели возможность просматривать его содержимое.
2. Обычная – проверка подлинности, которая требует имя пользователя Windows и пароль для доступа к содержимому. Учётная запись пользователя может быть локальной (на FTP-сервере) или находиться в домене. При обычной проверке подлинности через сеть передаются незашифрованные пароли. Эту проверку подлинности следует использовать, только если известно, что используется без-

опасное подключение между клиентом и сервером (с применением SSL).

К настраиваемым методам относятся:

1. Проверка подлинности «ASP.NET» – это метод проверки требует имя пользователя .NET и пароль для доступа к содержимому. Соответствующая учётная запись должна находиться в базе данных пользователей ASP.NET.
2. Проверка подлинности «Диспетчер IIS» – этот метод проверки использует учётные записи из диспетчера web-сервера. При этой проверке подлинности через сеть передаются незашифрованные пароли. Проверку подлинности «Диспетчер IIS» следует использовать, только если известно, что используется безопасное подключение между клиентом и сервером (с применением SSL).

После выбора необходимого метода проверки подлинности следует перейти к настройке параметров авторизации FTP (компонент «Правила авторизации FTP» (рис. IV.9.). На этой странице прописывают разрешающие и/или запрещающие правила пользователей и/или групп FTP-сервера (рис. IV.9.).

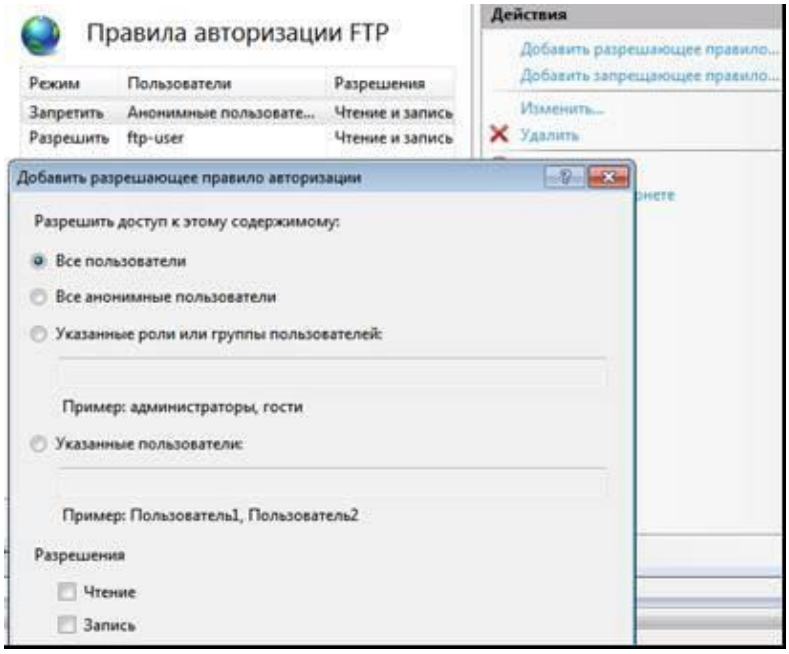


Рис. IV.9. Добавление правил авторизации FTP-сервера

Осталось настроить разрешения файловой системы NTFS для доступа к каталогам FTP-сервера его пользователей и/или групп. Для пользователей и/или групп FTP-сервера на соответствующие папки следует установить разрешения для выполнения манипуляций с их данными («запись», «чтение», «список содержимого папки» и т.д.).

Теперь FTP-сервер готов к работе.

Имена «FTP» и «Anonymous» зарезервированы системой. Невозможно создать учётные записи пользователей Диспетчер IIS, содержащие эти слова.

Информация по сессиям FTP-клиентов собирается IIS-службой и записывается в специальный журнал. Журнал помогает разрешать некоторые технические проблемы по работе FTP-сервера. Местоположение журналов можно узнать, выбрав компонент настройки «Ведение журнала FTP» (рис. IV.10) в окне «Диспетчер служб IIS» (рис. IV.8.).

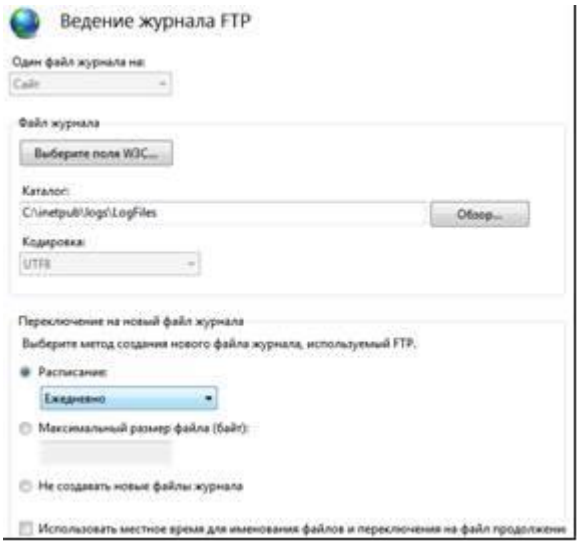


Рис. IV.10. Страница настройки журнала для FTP-сервера

В настройках «Диспетчер служб IIS» (рис. IV.8.) есть полезная функция «Текущие сеансы FTP». С помощью этой функции можно посмотреть текущие соединения и отключить при необходимости пользователей.

Функции компонента «Ограничения IPv4-адресов и имён домена» позволяет определять правила, разрешающие или запрещающие доступ к содержи-

мому FTP-сайта для определённого IPv4-адреса, диапазона IPv4-адресов или для имён домена, а также управлять этими правилами.

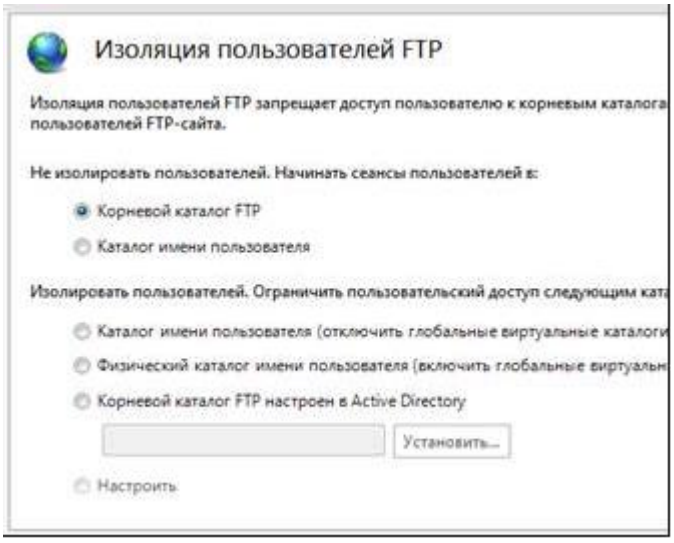


Рис. IV.11. Страница «Изоляция пользователей FTP»

Функция «Изоляция пользователей FTP» (рис. IV.11) позволяет управлять доступностью каталоги для пользователей и групп. С использованием данной настройки можно запретить просмотр или переписывание пользователями содержимого каталогов других пользователей за счёт предоставления доступа только к собственным каталогам. Пользователь не может подняться выше по дереву каталогов, поскольку его каталог верхнего уровня отображается как корень службы FTP. Только в пределах собственных каталогов пользователи могут создавать, изменять и удалять файлы или папки.

Безопасный FTP

Данные, которые передаются по протоколу FTP, как было отмечено ранее, не шифруются, поэтому имена пользователей, пароли, команды и другая информация могут быть перехвачены злоумышленниками. По этой причине были разработаны несколько методов, позволяющих зашифровать передаваемую посредством FTP-протокола информацию. Такие способы получили обобщающее название «Безопасный FTP». Наиболее распространёнными методами

защищённого обмена информацией между двумя компьютерами являются криптографические протоколы:

1. SecureShell (SSH, безопасная оболочка).
2. Transport Layer Security (TLS, защита транспортного уровня) и Secure Sockets Layer (SSL, уровень защищённых сокетов).

Эти криптографические туннельные протоколы разработаны для обеспечения безопасного конфиденциального обмена данными и связи по сети (особенно в сети Internet). Используемые в них технологии шифрования очень надёжны, при качественных настройках их практически невозможно взломать. Выделим основные безопасные FTP способы, а также протоколы, которые обеспечивают эту безопасность:

1. SFTP (SSH FTP) – протокол прикладного уровня для передачи файлов, который работает поверх безопасного канал. SFTP – это независимый протокол (не является надстройкой над FTP), который снабжён безопасной оболочкой SSH (SecureShell). SFTP шифрует абсолютно все команды, имена пользователей, пароли и другую конфиденциальную информацию. Этот протокол позволяет возобновлять передачу файлов, которая прервалась из-за проблем со связью.

Так как это совершенно другая конфигурация, клиенты, которые могут работать только с FTP, не могут соединиться с SFTP-сервером.

Другой протокол SimpleFileTransferProtocol имеет такую же аббревиатуру, что и протокол SSH FTP, но их нельзя путать.

2. FTP через SSH – создаёт обычную FTP-сессию через SSH-туннель, этот вариант нельзя путать с SFTP, который является отдельным независимым протоколом.

На сегодня Unix-подобные операционные системы (как Mac OS X и Linux) по умолчанию используют OpenSSH. Таким образом, программы или подсистемы, основанные на протоколе SSH, будут работать автоматически, без необходимости выполнять дополнительные процедуры установки, вносить су-

щественные поправки, создавать сертификаты для определённых видов безопасной передачи данных.

При использовании данного способа у пользователей операционной системы Windows может возникнуть проблемная ситуация, при которой данные клиента не будут защищены. Дело в том, что если несколько SSH-клиентов устанавливают туннель для управляющего канала, который изначально осуществляется через 21 порт (на практике такая ситуация встречается часто), то защищённым окажется именно этот канал. Напомним, что в пассивном режиме по 21 порту происходит входящее соединение, затем клиент и сервер используют механизм трёхстороннего рукопожатия, договариваются, по какому порту будет происходить обслуживание и переключаются на него. Поэтому при передаче данных клиентское программное обеспечение откроет новые TCP-соединения, которые будут находиться уже вне воздействия защитной оболочки SSH.

3. FTPS (FTP + SSL) – расширение стандартного протокола передачи файлов FTP (криптографическая надстройка данного протокола), которое обеспечено криптографическим протоколом SSL и его современной версией TLS, который появился позднее и основан на спецификации SSL 3.0. При использовании данных протоколов следует выделить два метода предоставления безопасности.

3.1. Неявный является устаревшим и использует стандартный протокол, требующий применения SSL или TLS, которые могут обеспечить шифрование информации. При таком методе обязательно нужно использовать порты, отличные от обычных, что создаёт неудобства, поскольку нарушается совместимость клиентов и серверов, не поддерживающих FTPS.

3.2. Явный – намного более удобный, так как применяет команды стандартного протокола FTP, но при отправке ответа зашифровывает информацию, что позволяет сохранить совместимость, поскольку в этом случае применяются одни и те же порты как для FTPS, так и для FTP. При этом для шифрования данных клиентом отправляется специальная команда.

В процессе настройки FTP-сервера (рис. IV.4.) есть возможность выбора SSL-протокола для шифрования передаваемых данных (можно выбрать «Разрешить» или «Требовать»). Протоколы FTPS и SFTP просты в использовании, но обеспечивают гораздо более высокий уровень защиты. Протокол FTPS, в отличие от SFTP, требует дополнительных действий для развёртывания – создания сертификата SSL, он более удобен, поскольку даёт возможность пользоваться обычными портами для передачи данных. Какой из «безопасных FTP» выбрать, зависит от вида задачи, которая поставлена перед сервером.

Установка и настройка FTP-сервера для ОС Linux

Рассмотрим вопросы установки FTP-сервера на операционной системе Linux Ubuntu. Будем использовать FTP-сервер vsftpd или VerySecure FTP Daemon, который, по мнению специалистов, обеспечивает высокий уровень защиты от уязвимостей.

Следует заметить, что большинство команд выполняется с привилегиями root.

В первую очередь необходимо обновить список пакетов в репозиториях, затем выполнить установку сервера:

```
$ sudo apt update
$ sudo apt install vsftpd
```

После завершения процесса установки можно включить сервис vsftpd, поскольку он не будет запущен по умолчанию, а также, при необходимости, следует добавить эту службу в автозагрузку:

```
$ sudo systemctl start vsftpd
$ sudo systemctl enable vsftpd
```

Если на хосте, где установлен и запущен сервер, работает Firewall (ОС Ubuntu Server), тогда для работы сервера необходимо открыть порты 20 и 21:

```
$ sudo ufw allow 20/tcp
$ sudo ufw allow 21/tcp
$ sudo ufw status
```

После завершения непосредственно процесса установки сервера, следует настроить его для безопасной работы.

Следует настроить vsftpd на использование криптографического протокола TLS, чтобы передаваемые по сети данные были зашифрованы.

Рекомендуется скопировать оригинальный файл настроек vsftpd.conf FTP-сервера, чтобы в случае проблемы вернуть настройки в начальное положение:

$ sudo cp /etc/vsftpd.conf /etc/vsftpd.conf.orig

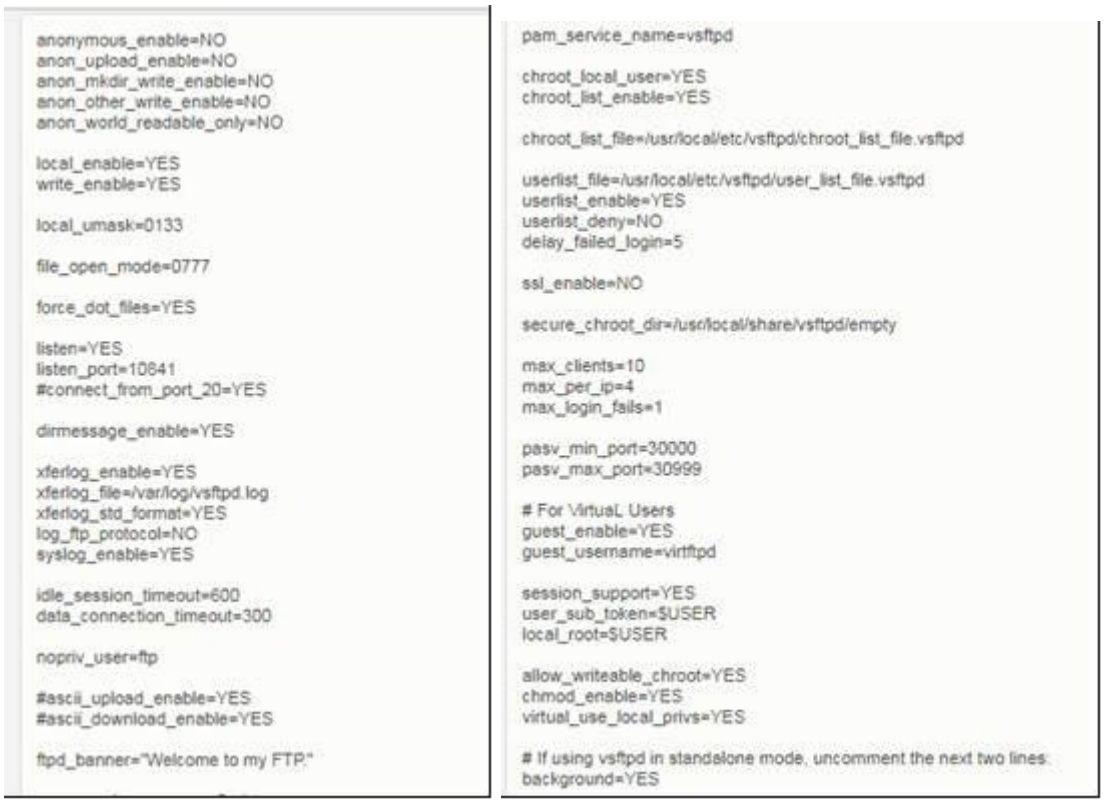


Рис. IV.12. Пример конфигурационного файла vsftpd.conf

Затем можно открыть файл в текстовом редакторе и изменить соответствующие строки (рис. IV.12.):

$ sudo vi /etc/vsftpd.conf

Отключаем анонимный вход:

anonymous_enable = NO

Разрешаем использовать имена локальных пользователей для входа:

local_enable = YES

Для авторизованных пользователей разрешаем команды, позволяющие изменять файловую систему:

write_enable = YES

Можно установить значение umask для новых файлов, создаваемых по FTP:

local_umask = 022

Получаем права для нового файлов – 644 (rw-r--r--) файла, права для директории – 755(rwxr-xr-x).

Включаем сообщение о необходимости выбрать каталог после регистрации:

dirmessage_enable = YES

Указываем, что следует записывать в логфайл все транзакции по передаче файлов.

xferlog_enable = YES

По необходимости можно указать использование стандартного формата для логфайла:

xferlog_std_format=YES

Разрешаем соединение через 20 порт:

connect_from_port_20 = YES

Переключаем сервер в режим ожидания соединений:

listen=YES

Регламентируем использовать PAM-сервис для установленного FTP-сервера vsftpd:

pam_service_name=vsftpd

PAM (Pluggable Authentication Modules) – это способ, позволяющий системному администратору задавать единые правила аутентификации для приложений и сервисов без перекомпиляции программ аутентификации.

PAM предоставляет множество полезностей системному администратору, таких как:

- Общая схема аутентификации, которая может быть использована разными приложениями (в том числе и FTP-сервером).
- С PAM могут работать разнообразные приложения, которые не нуждаются в перекомпиляции для поддержки PAM.
- Гибкость управления аутентификацией для администраторов и разработчиков ПО.
- Разработчикам не нужно привязывать свои приложения к конкретной схеме аутентификации. Вместо этого они могут сосредоточиться непосредственно на разработке самого приложения.

Можно разрешить аутентификацию только пользователей, перечисленных в файле userlist (в файле должны хранятся имена пользователей, которым запрещена аутентификация):

userlist_enable = YES

По умолчанию указанным в файле пользователям запрещён вход в систему, чтобы получить обратное: следует добавить следующую строчку:

userlist_deny=NO

Пользователям при входе отображается корневой каталог FTP-сервера. По необходимости можно ограничить возможности пользователей, разрешив им доступ только в своём (домашнем) каталоге. Для этих целей следует «активировать» строки («размещение пользователя в изолированном домашнем каталоге», «разрешена запись в домашний каталог»):

chroot_local_user = YES

allow_writeable_chroot = YES

После завершения установки следует сохранить изменения в конфигурационном файле и перезапустить сервер:

$ sudo systemctl restart vsftpd

Защита FTP-сервера (vsftpd) на основе TLS/SSL-сертификатов

Безопасность передачи данных на основе TLS/SSL-сертификатов обеспечивается при помощи аутентификации и шифрования. Прикладной протокол «заворачивается» в TLS/SSL, а тот в свою очередь в TCP/IP. Получается, что

данные по прикладному протоколу передаются по TCP/IP, но они зашифрованы. И расшифровать передаваемые данные может только тот клиент, который установил соединение, остальные не смогут их расшифровать. Шифрование выполняется ассиметричными алгоритмами.

Алгоритмы шифрования можно разделить на симметричные и асимметричные. При симметричном шифровании используется один ключ как для шифрования, так и для дешифрации. Алгоритмы этой категории гораздо быстрее асимметричных, но менее надёжные.

В асимметричных алгоритмах используются два ключа: открытый и закрытый. Шифрование производится с использованием открытого ключа, притом расшифровка этих данных может быть осуществлена только с использованием закрытого ключа. Зашифровать данные можно с помощью открытого ключа, однако, чтобы его расшифровать нужен только закрытый ключ. Если у пользователя нет закрытого ключа, то он не сможете расшифровать зашифрованный файла. Фактически зашифровать сообщение может каждый, но расшифровать его способен только владелец закрытого (секретного ключа).

При взаимодействии клиент-сервер, серверу заранее нужно подготовить ключи (открытый и закрытый). Открытый ключ должен быть подписан специальным Центром Сертификации (Comodo, LetsEncrypt), чтобы в дальнейшем гарантировать клиенту, что это именно тот сервер (домен, служба) за который он себя выдаёт. Подпись открытого ключа авторитетным центром гарантирует, что этот ключ подлинный и обслуживает именно тот домен (организация, сервис), для которого ключ и был создан. Открытый подписанный ключ копирует клиент, который будет устанавливать связь с этим сервером, а закрытый ключ хранится на сервера в приватном месте.

Если пользователь смог расшифровать отправленный файл или сообщение, зашифрованный с помощью своего открытого ключа, то значит – он владелец. Ключи работают только в паре.

Асимметричные алгоритмы работают гораздо медленнее, но надёжнее симметричных. Они предназначены для шифрования небольшого объёма данных.

Подписать открытый ключ можно через доступные онлайн-сервисы, которым можно отправить свой ключ, заполнив соответствующую анкету. Далее сервис (центр сертификации) обрабатывает данные из анкеты и сам ключ и высылает уже подписанный ключ обратно его владельцу.

Между клиентом и сервером используются следующая схема взаимодействия, на основе асимметричных алгоритмов:

1. Клиент устанавливает соединение с сервером и запрашивает защищённое подключение.

2. При установке соединения клиент предоставляет список доступных ему алгоритмов шифрования. Сервер из списка выбирает приемлемый алгоритм, после чего сообщает клиенту, какой алгоритм может использоваться.

3. Сервер отправляет клиенту свой цифровой сертификат, подписанный удостоверяющим центром, и открытый (публичный) ключ.

4. Клиент может связаться с сервером Центра Сертификации, который подписал сертификат сервера, и проверить, соответствует ли сертификат данному серверу (домену). Но, может и не связываться, так как в ОС обычно уже установлены сертификаты Центров Сертификации.

5. Клиентом генерируется сеансовый ключ (случайная последовательность) для защищённого соединения.

6. Далее клиент зашифровывает этот ключ посредством асимметричного алгоритма, используя известный ему заранее открытый ключ сервера. Зашифрованный ключ отправляется серверу.

7. Сервер получает и расшифровывает полученный ключ (закрытым ключом), сообщает клиенту, что все хорошо.

8. Дальнейший трафик между сервером и клиентом шифруется посредством выбранного симметричного алгоритма. Устанавливается зашифрованное

соединение. Данные, передаваемые по нему, шифруются и расшифровываются до тех пор, пока соединение не будет разорвано.

Под SSL-сертификатом обычно понимают открытый и закрытый ключ, а также подпись открытого ключа. В процессе создания SSL-сертификата происходит последовательная обработка следующих видов ключей:

*.key – это сами ключи шифрования, открытий и/или закрытый;

*.csr – ключ, содержащий Клиента, генерирует случайную цифровую последовательность, сформированный запрос для получения подписи сертификата от центра сертификации, а сам запрос – это открытый ключ и информация о домене и организации, связанной с ним;

*.crt, *.cer, *.pem – это, собственно, сам сертификат, подписанный центром сертификации по запросу из файла *.csr.

Следующим этапом следует позаботиться о шифровании данных передаваемых между пользователями и FTP-сервером (vsftpd) на основе сертификатов TSL/SSL. Для этого необходимо выполнить следующие шаги:

1. Генерация закрытого ключа.
2. Создание запроса на подпись открытого ключа.
3. Подпись сертификата (открытого ключа).
4. Внесение SSL в конфигурацию FTP-сервера (vsftpd).

Генерация закрытого ключа:

$ openssl genrsa -des3 -out vsftpd.key 1024

Команда genrsa генерирует RSA-ключ, опция -des3 указывает алгоритм шифрования ключа, опция -out указывает имя файла-ключа vsftpd.key, число 1024 – это сложность шифрования. В процессе выполнения данной команды будет предложено ввести пароль для шифрования.

Создание запроса (csr-запрос) на подпись открытого ключа, который будет включать только что сгенерированный ключ vsftpd.key:

$ openssl req -new -key vsftpd.key -out vsftpd.csr

При выполнении этой команды пользователю необходимо ввести информацию о домене и организации. Получить подписанный сертификат, когда име-

ется закрытый ключ и запрос на подпись, можно несколькими способами: воспользоваться услугой авторитетных центров сертификации (отправив им запрос: vsftpd.csr) и получив в ответ готовый сертификат vsftpd.crt. Либо можно сделать сертификат самоподписанным. Т.е. подписать его тем же ключом, который использовался при создании файла vsftpd.csr. Для этого следует выполнить команду:

$ openssl x509 -signkey vsftpd.key -in vsftpd.csr -req -days 365 -out vsftpd.crt

Опция -days указывает количество дней, в течение которых выпускаемый сертификат vsftpd.crt будет действителен.

Осталось перенести сертификат в соответствующую папку и изменить конфигурационный файл FTP-сервера.

Можно объединить команды по созданию ключа, запроса, «самоподписи» сертификата (сразу следует указать каталоги для хранения файлов SSL-сертификата):

$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout /etc/ssl/private/vsftpd.key -out /etc/ssl/private/vsftpd.crt

Этой командой служба OpenSSL создаст «самоподписанный» сертификат, действительный на протяжении года. Файлы создали в каталоге /etc/ssl/private/. Теперь следует изменить конфигурационный файл (с привилегиями root):

$ sudo vi /etc/vsftpd.conf

Следует добавить строки с указание закрытого ключа и сертификата SSL

rsa_cert_file=/etc/pki/tls/certs/mycertificate.crt
rsa_private_key_file=/etc/ssl/private/vsftpd.key

Следующие строки позволяют ограничить доступ клиентов сервера к SSL-сертификату.

ssl_enable=YES
allow_anon_ssl=NO
force_local_data_ssl=YES
force_local_logins_ssl=YES

Можно настроить FTP-сервер на использование TLS (современная версия SSL):

```
ssl_tlsv1=YES
ssl_sslv2=NO
ssl_sslv3=NO
require_ssl_reuse=NO
ssl_ciphers=HIGH
```

После сохранения изменений в конфигурационном файле, следует перегрузить сервер для активации изменений.

Выше рассмотрена установка и настройка FTP-сервера на основе криптографического протокола (FTPS). FTPS использует несколько номеров портов при подключении клиентов к серверу, поэтому для брандмауэра управление такими подключениями является трудоёмким процессом.

Лучшим решением в этом случае является использование SFTP (FTP через SSH). SFTP использует только порт 22.

Этот порт используется для всех подключений во время сеансов FTP.

Если вы используете брандмауэр, то рекомендуем выбрать именно SFTP, так как ему нужен только один порт при подключении.

Некоторые рекомендации по настройке SFTP-сервера для ОС Linux

Для установки SFTP-сервера достаточно установить на хосте OpenSSH-server (если он ещё не установлен), на стороне клиента требуется установить пакет SSH. Команда для установки OpenSSH-server:

```
sudo apt install openssh-server
```

Установим и SSH в этой же системе, чтобы в дальнейшем проверить работу SFTP-сервера (как клиент получит доступ к SFTP-серверу).

```
$sudo apt install ssh
```

Теперь следует перейти к настройке SFTP-сервера. Настройка выполняется за несколько шагов:

1. Создание группы пользователей для работы на сервере (добавление в эту группу пользователей). Например, пользователь sftpuser в группе sftpgroup.

2. Создание каталогов, изменение их прав доступа и владельца таким образом, чтобы при подключение пользователю был доступен именно его (домашний) каталог, а не корневой. Для удобства можно дополнительно реализо-

вать правило для пользователей: они могут читать файлы из домашнего каталога, но загружать могут только в подкаталог загрузки.

```
sudo mkdir -p /sftpdir/ftpuser/loaddir
sudo chown -R root.sftpgroup /data/ftpuser
sudo chown -R sftpuser.sftpgroup /data/ftpuser/loaddir
```

3. Настройка конфигурационного файла (sshd_config).

Следует добавь следующие строки в /etc/ssh/sshd_config:

```
Match Group sftpgroup
ChrootDirectory /sftpdir/%u
ForceCommand internal-sftp
```

Зарегистрированный пользователь из группы sftpgroup при подключении к серверу попадает в свой каталог и не может видеть ничего выше этого каталога (доступ к файлам других пользователей закрыт).

4. Перегрузка сервера после изменения корневого файла

```
sudo systemctl restart sshd
```

5. Проверка работы установленного сервера (подключение через командную строку)

```
$sftp sftpuser@127.0.0.1
```

Если сервер корректно работает, а пользователь успешно прошёл аутентификацию, то ему будет доступен домашний каталог /sftpdir/ftpuser.

Установка и настройка SFTP-сервера для Windows

Как уже было отмечено, чтобы установить и настроить SFTP-сервер нужен SSH, во всех версиях Windows данной службы нет (только Windows 10 и Windows Server 2019), поэтому необходимо устанавливать дополнительное программное обеспечение, например, OpenSSH.

Настройка SFTP-сервер для ОС Windows включает:

1. Установка и настройка OpenSSH.
2. Открытие порта для работы SFTP.

Установка и настройка OpenSSH

1. Загрузите и распакуйте архив OpenSSH в папку Program Files.

2. Откройте командную строку (командой powershel.exe) от имени администратора и перейдите к папке C:\Program Files\OpenSSH-Win64.
3. Для работы SSH требуются две службы: sshd (сервер SSH – OpenSSH) и ssh-agent (агентом проверки подлинности OpenSSH). Для установки этих служб следует выполнить команду: powershell.exe -ExecutionPolicy Bypass -File install-sshd.ps1.

   Это команда создаст папку C:\ProgramData\ssh с ключом хоста, который имеет важное значение для работы службы.
4. После установки OpenSSH, проверьте работоспособность данного сервиса.

Можно всякий раз запускать эти службы вручную или настроить их автоматическое открытие с помощью Services.msc.

Открытие порта для работы SFTP

SSH использует 22 порт для передачи любых данных, а этот порт «по умолчанию» в ОС Windows закрыт. Чтобы открыть порт, следует в Wirewall (брандмауэре Windows) настроить «Разрешить подключение»:

Панель управления >Администрирование->Брандмауэр Windows в режиме повышенной безопасности->Правила для входящих подключений (на левой панели)-> Создать правило .

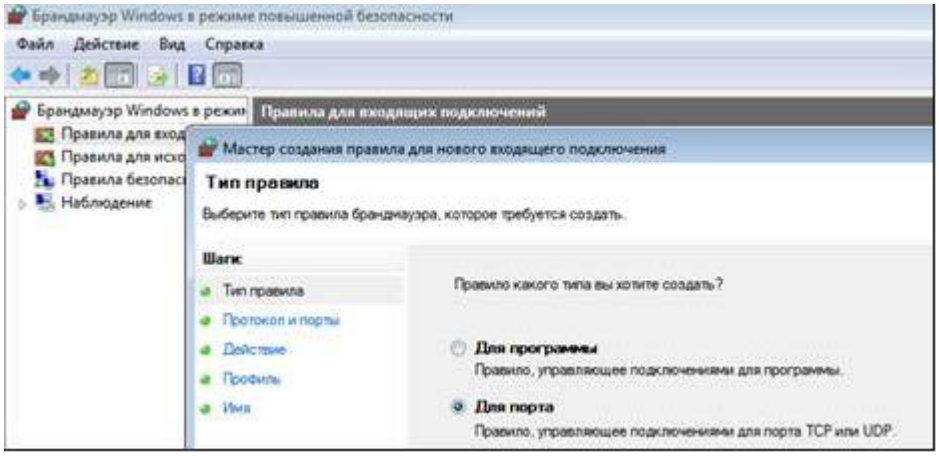


Рис. IV.13. Мастер создания правила для нового входящего подключения

Далее необходимо следовать инструкциям «Мастера создания правила для нового входящего подключения» (рис. IV.13.):

- тип правила – «для порта»;
- протокол и порты – «определённые локальные порты»: 22 порт;
- действие – «Разрешить подключение»;
- профиль – «Частный»;
- имя – SSH или OpenSSH.

Для проверки работоспособности SFTP-сервера нужен SFTP-клиент. Можно использовать WinSCP, FileZilla, FTP Rush или файловый менеджер с поддержкой SFTP, такой как Altap Salamander.

V. Основы безопасной разработки web-приложений

В данном разделе мы рассмотрим особенности разработки web-приложений с точки зрения информационной безопасности. Большинство, примеров и рекомендаций, приведенных в этом разделе, будут актуальны не зависимо от того, какими платформами и инструментами вы пользуетесь при разработке web-приложений. Однако для наглядности мы будем рассматривать данные примеры с использованием языка PHP. Выбор в пользу этого языка был сделан не случайно, так как на момент написание данного пособия он является самым популярным в мире языком программирования специализирующемся на разработке web-приложений и сервисов.

К основным факторам, влияющим на безопасность web-приложений без привязки к какому-либо языку программирования или фреймворку, можно отнести два группы фактора. Первое, ответственное использование программно-аппаратного комплекса актуального с точки зрения разработки и эксплуатации Web приложений. Второе, корректная работа со всеми данными, приходящими от пользователя.

К первой группе факторов мы отнесем использование надежного программного обеспечения, имеющего положительную репутацию, его своевременное обновление и адекватную настройку конфигурации.

Вторая группа факторов связана с особенностью эксплуатации web-приложений, при которой разработчик не может доверять никаким клиентским данным, приходящим в результате GET, POST запросов, загружаемых файлов, cookie и т.д. Вся информация, приходящая от пользователя, требует критического отношения и проверки, как на корректность, так и на подлинность.

V.1. Обработка данных форм

При разработке web-приложения нужно быть готовым к тому, что пользователь может ввести в форму на странице любую информацию, а не только ту

информацию, которую мы от него ждем. Этом может быть как не преднамеренная ошибка, когда из-за смены фокуса и невнимательности пользователя он введет в поле для ввода имени, номер своего телефона. Так и преднамеренное вредоносное воздействие, когда пользователь может попытаться провести атаку внедрения кода.

Достаточно часто встречается такая ситуация, когда web-разработчик желая обезопасить свое приложение, разрабатывает различные фильтры препятствующие ввести недопустимые данные в поле формы. Однако следует понимать, что злоумышленнику для передачи данных в скрипт не обязательно использовать исходную форму. В случае с GET – запросами, он может провести необходимые манипуляции с переменными в адресной строке браузера. В случае с POST запросами он может модифицировать форму или же попытаться создать свою. Поэтому проверка данных формы на стороне браузера должна реализовываться с точки зрения удобства пользователя. Без перезагрузки страницы напомнить ему о необходимости заполнения какого-то поля, или же указать на неправильный формат данных и т.д. Проверка же данных должна происходить на стороне сервера. Это, кстати в равной мере, касается и приходящих от пользователя данных, сессий, cookies, файлов и т.д.

Какую же проверку необходимо провести с данными, поступившими из формы в скрипт. Начнем с переменных. Каждая поступающая переменная должна быть проверена:

1. На существование – выполнение той или иной части скрипта может быть начато только в том случае если все необходимые переменные существуют. При этом достаточно часто может возникнуть ситуация, когда переменная может не являться критической и не влиять на исполнение логики скрипта. Например, добавление какого-то текстового поля с описанием в таблицу базы данных. В этом случае рекомендуется, использовать значение по умолчанию. То есть убедившись, что какая-то переменная не поступила в запрос, и понимая, что это переменная не является критичной для неё будет использовано

значение по умолчанию. Это поможет предотвратить ситуации, связанные с подменной переменной в дальнейшем.

2. На тип данных – переменные должны быть приведены к необходимому виду. Следует обратить внимание, что сегодня большинство языков программирования применяющихся при web-разработке являются слабо типизированными и допускают использования переменных без объявления типа. Однако в целях безопасности, если для переменной не был объявлен тип рекомендуется все же указать интерпретатору, с чем он может столкнуться. Часто для этого можно использовать помимо традиционных методов, кавычки для строковых переменных или умножение на 1 для числа.

3. На использование допустимой кодировки – в ряде случаев злоумышленник может сгенерировать запрос, который будет содержать инъекцию типа внедрения кода, при этом использование некорректной кодировки может обмануть средства по фильтрации недопустимого содержимого.

4. На использование допустимых символов – нужно удостовериться что переменная не содержит запрещённые символы. Например, кавычки, инъекции, тэги и т.д.

V.2. Обработка загрузки файлов

Потенциально загрузка файлов может быть использована злоумышленником для реализации сразу нескольких типов атак, начиная от примитивной атаки на свободное дисковое пространство и заканчивая загрузкой на сервер бэкдоров для удаленного управления сервером. Именно поэтому загрузка файлов должна быть строго регламентирована как с точки зрения настройки самого web-сервера, так и с точки зрения разработки web-приложения.

Рассмотрим возможные угрозы. Одним из самых простых способов вызвать отказ сервера является атака на свободное дисковое пространство. Этот вид атака является одним из-под видов семейства DDOS-атак. Злоумышленник может инициировать несколько процессов загрузки файлов и полностью утили-

зировать свободное пространство, что скорее всего вызовет отказ всего сервера. Какие способы противодействия таким атакам могут быть.

Ограничение объема загружаемого файла. Установить такое ограничение можно несколькими способами. Самый простой из них, но при этом и самый не надежный, это указать ограничение в самой форме передачи файла используя тэг <INPUT> с name="MAX_FILE_SIZE". Однако это ограничение может быть легко преодолено злоумышленником, например, путем модификации формы или размещения формы на другом ресурсе. Другой способ, ограничения на объем загружаемого файла, это ограничения его на стороне сервера. Например, при использовании PHP, разработчик может указать в конфигурационном файле php.ini параметр upload_max_filesize, указывающий максимально допустимый объем загружаемого файла. Однако в этом случае следует помнить, что заданное значение не должно быть больше параметра post_max_size, который ограничивает в целом все данные, переданные в запросе методом POST. Ещё один способ контролировать и ограничивать размер это проверять его в самом скрипте при загрузке файла контролируя размер через переменную $_FILES['userfile']['size']. При разработке функционала загрузки файлов целесообразно использовать все эти инструменты. Так, например, ограничение на стороне браузера не спасет от злоумышленника, но будет полезно легитимному пользователю, который сразу увидит ограничение, а не будет дожидаться окончание загрузки файла, чтобы увидеть сообщение об ошибке. Ограничение на уровне сервера позволят отбросить файл до его передачи в скрипт, чем сэкономит ресурсы сервера, и наконец проверка в скрипте позволит точно установить размер файла и при необходимости прервать его обработку.

Также полезной рекомендацией будет размещение библиотеки загружаемых файлов на отдельном носителе, а не на системном разделе. В этом случае, даже если все свободное место будет утилизировано, отказа сервера не произойдет.

Если существует реальная угроза переполнения дискового пространства пользовательскими файлами. Можно ввести какие-то дополнительные ограни-

чения, например, на количества передаваемых файлов в сутки или на время между двумя загрузками файла. Для реализации этого функционала потребуется на стороне сервера вести учет загружаемых файлов.

Вторая наиболее распространённая угроза, это попытка через интерфейс загрузки файлов передать на сайт файл, содержащий бэкдор, по сути, скрипт, который позволит, например, управлять сервером, эмулировать консоль и т.д. Здесь также есть несколько способов противодействия. Это ограничение на тип передаваемых данных, на расширение и модификация файла.

Ограничение на тип передаваемых файлов может быть легко преодолено злоумышленником путем подделки. Ограничение на расширение имени файла является более продуктивным, но также может быть преодолено. Говоря о расширении, мы имеем в виду две важные вещи. Первая – это обработку файлов только с определенным расширением, второе это замену пользовательского названия, но с указанием правильного расширения. Например, если пользователь передает нам файл image.jpg мы проверяем расширение на соответствие нашим правилам и сохраняем файл с новым названием и правильным расширением. Например, вот в таком виде $user_id.”_”.time().”.jpg”. Такое название файла позволит установить нам владельца файла и время загрузки. Что может быть использовано, например, для учета загружаемых файлов без использования базы данных.

Однако и этот способ не является гарантией того, что злоумышленник не сможет загрузит на сервер вредоносный код и заставить его исполниться. Примером, может быть web-приложение, которое использует PHP для создания и обработки изображений. В этом случае велик шанс, что файлы с расширением JPG могут обрабатываться интерпретатором. Чтобы избежать таких случаев и для максимальной безопасности можно воспользоваться одной уловкой – модифицировать файл. Например, для изображения, прежде чем сохранить его в целевое место, можно выполнить функцию resize(), если изменение размера изображения не входит в наши задачи, это может быть приведение размера к существующему, для офисных документов это может быть дополнение какой-

то информации в мета данные документы и т.д. Главный смысл этого способа, заключается в том, что при обработке поступившего файла целевым способом, если это легитимный файл, ему не будет нанесен какой-либо урон. Если же это файл вредоносный, его содержимое будет либо «обезврежено», либо его обработка закончится ошибкой.

V.3. Аутентификация

Далее, рассматривая основы безопасной разработки web-приложений необходимо поговорить об аутентификации пользователей. И эта часть также будет связанна с данными, которые нам передает пользователь.

Начать здесь нужно со стойкости паролей. Это извечная проблема, так как с одной стороны, пользователи стремятся использовать как можно более простые пароли, с другой стороны, злоумышленники делают всевозможные атаки на средства аутентификации.

Начать решение этих проблем стоит с введения определённых требований к паролям. Чтобы они были определенной длинны, имели цифры, строчные и прописные буквы и т.д. И ограничить попытки проведения аутентификации для того, чтобы осложнить злоумышленникам процедуру подбора пароля.

Однако ограничиваться этими мерами не стоит, так как всегда существует риск, скомпрометировать базу данных логинов и паролей в случае какого-либо инцидента информационной безопасности. Например, компрометация базы – в результате sql-инъекции. Следующем шагом в разработке безопасной аутентификации должно стать хранение всех паролей в зашифрованном виде. Причем желательно без возможности обратного расшифрования, например, в виде хеша пароля. Отсутствие обратного расшифрования не помешает проверке подлинности пользователя, так как его, поступивший в процессе аутентификации пароль, можно шифровать и сравнивать между собой сохранённый в базе данных хеш и хеш который поступил из формы аутентификации.

Хранение паролей в зашифрованном виде, затруднит доступ к ним злоумышленника, но все же не станет абсолютной гарантией безопасности. Так как

злоумышленник имея значительные вычислительные мощности может взломать пароли путем перебора или же уже имея всевозможные хеши паролей из словарей просто сравнить полученные хеши.

Для того, чтобы минимизировать данную угрозу рекомендуется использовать, так называемый, соленный хеш. То есть, при создании хеша пароля, добавить к нему соль. По сути, дополнительную текстовую строку, которая изменит хеш пароля и затруднит его перебор. Например, если пользователь использует пароль вида 1q2w3e, добавление к этому паролю строки «sd6fsd2br78ew78f98776biwsndafh», которая будет называться солью, значительно затруднит злоумышленнику его подбор.

Для ещё большего увеличения стойкости хеша пароля, рекомендуется использовать применение, к получаемому хешу, ещё несколько функций хеширования, возможно разных, каждый раз добавляя новую соль. Это позволит значительно затруднить подбор пароля, особенно в тех случаях, когда злоумышленник, может зарегистрироваться в системе и видит свой собственный хеш. В этом случае он будет знать свой собственный пароль, и взлом базы данных с паролями сведется, на первом шаге, к перебору соли к своем личному паролю.

После прохождения процедур аутентификации сервер должен запомнить, что этот конкретный пользователь уже прошел процедуру аутентификации. Оптимально это сделать используя cookie. То есть, запомнить в браузере пользователя его статус. Естественно, создавая cookie необходимо помнить, что она может быть подделана злоумышленником или похищена у легитимного пользователя. Поэтому при реализации средства аутентификации рекомендуется:

1. Не использовать в cookies имя пользователя или его идентификатор, так как эта информация может быть подделана.
2. Вести на стороне учет назначенных cookies и IP-адресов. Это предотвратит возможность использовать похищенную cookie.
3. Своевременно уничтожать cookie, инициируя завершения сеанса.

VI. Программные средства для обеспечения безопасности web-сервера

VI.1. Web-брандмауэр

Web-сервер (web-server, интернет сервер) – это программное обеспечение, отвечающее за приём и обработку входящих сетевых запросов на web-ресурсы по протоколу HTTP. Также web-сервер выполняет функцию исполнения скриптов, например, таких как CGI, JSP, ASP и PHP, которые отвечают за организацию запросов к сетевым службам, базам данных, доступу к файлам, пересылке электронной почты и другим приложениям.

Web-серверы часто являются целями атак. Вследствие этого вопросам обеспечения безопасности web-серверов, а также сетевой инфраструктуры, которая их поддерживает стоит уделять внимание в первую очередь. Обеспечение безопасности web-сервера – это комплекс мер защиты от взлома, одной из которых является использование web-брандмауэра.

Web-брандмауэр (межсетевой экран для web-приложений, web Application Firewall, WAF) – это межсетевой экран для фильтрации трафика прикладного уровня, специально ориентированный на web-приложения. В основе лежит тот же принцип, что и в обычных брандмауэрах – контроль и анализ всех пакетов, поступающих от клиента. Чтобы обнаружить и предотвратить атаки WAF проверяет все запросы к web-серверу и соответствующие ответы сервера на соответствие своему набору правил. Web-брандмауэр строит сравнительную модель безопасности, которая описывает, как web-приложение работает при нормальных условиях. Затем он сравнивает реальные потоки данных с моделью, что позволяет ему идентифицировать необычное поведение. Эффективная модель приложения WAF включает в свой состав динамические URL (.asp, .php и т.д.), параметры, методы HTTP, маркеры (cookies), идентификаторы сеанса, схемы XML/SOAP и многое другое.

WAF функционирует на основе двух общепринятых моделей безопасности:

- Negative – отрицательная модель или чёрный список (отрицает то, что является заведомо установленным). Для предоставления базовой защиты WAF может использовать как общеизвестные сигнатуры для предотвращения наиболее распространённых атак, так и специфические сигнатуры для атак, использующих уязвимости отдельных WEB-приложений.
- Positive – положительная модель или белый список (разрешает только то, что является заведомо установленным). Для улучшенной защиты, в дополнение к сигнатурам, используется ещё один тип логики: правила, которые определяют, что явно разрешено.

WEB-брандмауэры разделяют на 2 типа: аппаратный и программный. Наибольшее распространение получила программная реализация WAF из-за более простой реализации.

По принципу действия WAF можно разделить на 3 типа:

1. Реализованные в виде обратного прокси-сервера. В данном типе WAF все данные сначала обрабатываются прокси-сервером, который уже решает, пропускать пакеты или блокировать. В случае положительного результата данные перенаправляются к web-серверу без изменения либо с частичной правкой.
2. Работающие в режиме маршрутизации/моста. К этому типу чаще всего относят аппаратные WAF. У данного типа реализации есть как плюсы, так и минусы. К первым можно отнести прирост производительности, ко вторым – более сложную и тонкую настройку.
3. Встроенные в web-приложения. Данный тип WAF встраивается непосредственно в web-приложение в качестве дополнительного функционала и работает на программном уровне.

Классическое размещение WAF в сети – в режиме обратного прокси-сервера, перед защищаемыми web-серверами. Одним из примеров таких

средств защиты является широко распространённый web-брандмауэр ModSecurity.

VI.2. Использование ModSecurity

ModSecurity – это межплатформенный брандмауэр web-приложений с открытым исходным кодом (выпущен под лицензией Apache 2.0), который подходит для защиты ряда различных платформ, включая Apache HTTP Server, Microsoft IIS и NGINX.

В ModSecurity имеется язык конфигурации правил SecRules ModSecurity, для мониторинга, регистрации и фильтрации сообщений по протоколу передачи гипертекста в режиме реального времени на основе пользовательских правил.

ModSecurity чаще всего разворачивается для обеспечения защиты от общих классов уязвимостей с использованием OWASP ModSecurity Core Rule Set.

OWASP ModSecurity Core Rule Set (CRS) – это набор общих правил обнаружения атак для использования с ModSecurity или совместимыми межсетевыми экранами WEB-приложений. Целью CRS является защита web-приложений от широкого спектра распространённых категорий атак, включая OWASP Top-10, с минимумом ложных предупреждений.

Для обнаружения угроз механизм ModSecurity встраивается в web-сервер или развёртывается в качестве прокси-сервера перед web-приложением. Это позволяет ядру ModSecurity сканировать входящие и исходящие сообщения HTTP-трафика и в зависимости от конфигурации правила решать, как следует обрабатывать сообщения: передавать, отбрасывать, перенаправлять, возвращать заданный код состояния, выполнять сценарий пользователя и многое другое.

Существует два варианта установки ModSecurity:

1. Установка непосредственно на web-сервер.
2. Установка в качестве обратного proxy-сервера.

В случае установки ModSecurity на web-сервер, брандмауэр перехватывает все запросы к web-серверу, на котором он установлен: запрос от клиента

сначала сравнивается с правилами фильтрации модуля ModSecurity, а затем передаётся на дальнейшую обработку web-серверу. ModSecurity также может контролировать ответы web-сервера, т.е. после формирования страницы ответ обрабатывается модулем и в соответствии с правилами блокирует или разрешает прохождение ответа web-сервера.

Установку в качестве обратного proxy-сервера можно использовать для замены страниц с ошибками web-приложений, возникающими в результате обработки web-сервером запросов от клиента, на стандартные страницы (например, страницы с ошибкой 404). Это позволяет уменьшить объем передаваемой клиенту критичной информации о работе приложения и усложняет задачу потенциального злоумышленника. Принцип работы обратного proxy-сервера заключается в том, что сервер переадресует все запросы от клиентов WEB-серверу и при получении ответа перенаправляет его клиенту. Клиенты «видят» только внешний интерфейс proxy-сервера и «не видят» расположенных за ним WEB-серверов. В данном случае недостатками являются необходимость закупки дополнительного оборудования под proxy-сервер и наличие единой точки отказа для одного или нескольких WEB-серверов. Однако сам proxy-сервер может выполнять и дополнительные полезные функции, например, кэширование статических запросов и балансировку нагрузки на группу WEB-серверов.

Рассмотрим основные возможностей, предоставляемых модулем ModSecurity:

1. Анализ запроса. Это основная функция данного модуля, особенно когда вы имеете дело с POST запросами, в которых получение тела запроса может быть затруднено.
2. Выполнение канонизации и функции антиуклонения. Выполнение ряда преобразований, для преобразования входных данных в форму, подходящую для анализа. Этот шаг применяется для борьбы с различными методами уклонения.
3. Выполнение специальных встроенных проверок. В этом месте выполняются более сложные проверки правильности, такие как про-

верка правильности URL кодирования и проверка правильности Unicode кодирования. Вы можете также контролировать некоторые значения байтов в запросе для борьбы с shellcode.

4. Запуск входных правил. В этом месте запускаются правила, созданные пользователем. Они позволяют анализировать каждый аспект запроса, используя регулярные выражения. Также здесь могут быть объединены несколько правил для более сложного анализа.

Также, кроме обычных ModSecurity соединений может контролировать и защищённый SSL-трафик.

VI.3. Контроль целостности

Одной из важных задач информационной безопасности является обеспечение целостности данных. Причём под целостностью понимается не только их не поврежденность, но и подлинность информации. Проблема заключается в том, что злоумышленник может, используя те или иные уязвимости в программном обеспечении серверов, попытаться исправить какие-то данные. В некоторых случаях эта угроза не несёт большой опасности. В других же случаях подмена информации может привести к действительно большим убыткам. Именно поэтому контроль целостности данных зачастую является важной задачей, для решения которой предназначено специальное программное обеспечение – подсистема контроля целостности данных.

Подсистема контроля целостности данных отвечает за подлинность информации. Алгоритм работы этой подсистемы выглядит следующим образом. Системный администратор указывает подсистеме контроля целостности список защищаемых файлов. Затем программа проверки целостности файла вычисляет и хранит криптографическую контрольную сумму каждого защищаемого файла и поддерживает базу данных контрольных сумм файлов. Контрольные суммы должны перевычисляться регулярно для сравнения текущего значения с хранимым значением, что позволяет определить любые модификации файла. При выявлении несоответствия между эталонным и полученным значением фиксиру-

ется факт искажения файлового ресурса, о чём немедленно оповещается администратор. Это позволяет системному администратору легко распознать изменения, в частности, неавторизованные, сделанные в критичных файлах.

Возможность проверки целостности файлов часто включается в системы обнаружения проникновений для хоста, но такая проверка может быть доступна и в виде отдельной программы.

Подсистема контроля целостности не является превентивным средством защиты, поскольку позволяет выявить лишь последствия информационного вторжения. Однако наличие такой подсистемы, по существу, необходимо поскольку, если все имеющиеся средства защиты пропустили информационную атаку, то подсистема контроля целостности позволяет выявить её последствия. Эти сведения служат для пользователей гарантией подлинности содержимого web-сервера и целостности системы безопасности подключения HTTP.

Хотя программа проверки целостности и является полезным инструментальным средством, которое не требует большого участия человека, для гарантии эффективности она должна использоваться продуманно.

При создании первого варианта базы данных файлом проверки целостности требуется гарантия, что система находится в безопасном состоянии. В противном случае могут быть созданы криптографические хеши скомпрометированной системы, и тем самым создастся ложное чувство безопасности у тестирующего.

База данных контрольных сумм должна храниться off-line, чтобы атакующий не мог скомпрометировать систему и модифицировать базу данных с целью спрятать следы атаки.

Программа проверки целостности файлов может также создавать ложные позитивные предупреждения. Каждое изменение файла и каждый системный patch изменяет файл и, следовательно, требуется изменить базу данных контрольных сумм. Таким образом, держать базу данных актуальной может быть непросто.

Большинство современных web-серверов, реализуют криптографический протокол Secure Sockets Layer (SSL), обеспечивающий поддержание конфиденциальности, целостности и подлинности передаваемых по сети данных. Стандарт SSL был разработан в 1994 году компанией Netscape, и с января 1999 года IETF стандартизирует его под именем TLS. В TLS-документации говорится, что «разница между этим протоколом и SSL 3.0 некритична». SSL и TLS представляют собой постоянно обновляемую серию протоколов, и их часто в обиходе объединяют в общее название SSL/TLS.

Безопасный обмен данными обеспечивается за счёт шифрования и аутентификации цифрового сертификата, представляющего собой файл, который уникальным образом идентифицирует web-серверы. Обычно цифровой сертификат подписывается и заверяется специализированными центрами сертификации (удостоверяющими центрами).

SSL-сертификат – это цифровая подпись web-сайта, подтверждающая его подлинность. Использование сертификата позволяет защитить как владельца web-сайта, так и его клиентов. SSL-сертификат даёт возможность владельцу применить к своему web-сайту технологию SSL-шифрования.

Информация передаётся в зашифрованном виде, и расшифровать её можно только с помощью специального ключа, являющегося частью сертификата. Тем самым гарантируется сохранность данных. Посетители web-сайта вправе ожидать, что защита их информации, если она важная, будет обеспечена с помощью SSL-сертификата. Если сайт имеет SSL-сертификат, то в строке состояния браузера отображается значок в виде замка.

Выделяют три ключа, которые используются для шифрования соединений:

1. открытый;
2. закрытый;
3. сеансовый.

Шифрование работает по следующему принципу: то, что зашифровано с помощью открытого ключа, может быть расшифровано с использованием за-

крытого ключа и наоборот. Автоматически создаётся симметричный сеансовый ключ и затем применяется для шифрования данных, передаваемых после создания безопасного соединения. Всякий раз, когда браузер обращается к защищённому web-сайту, устанавливается соединение.

Таким образом, назначение SSL-сертификата – обеспечить безопасное соединение между web-сервером и браузером пользователя, защитив данные от перехвата и подмены. Сертификат используется для шифрования данных и идентификации web-сайта при установлении защищённого соединения HTTPS. Посмотреть SSL можно, кликнув мышью в поисковой строке на замке перед HTTPS.

Рис. V.1. Схема действия SSL
(https://habr.com/ru/company/regru/blog/280878/ - Дата доступа 08.06.2020)

Рассмотрим алгоритм работы HTTPS (рис. V.1):

1. Пользователь заходит на защищённый web-сайт.
2. Выполняется проверка DNS и определение IP-адреса хоста web-сайта.
3. Запись web-сайта найдена, переход на web-сервер хоста.
4. Запрос безопасного SSL-соединения с хоста web-сайта.
5. Хост отвечает валидным SSL-сертификатом.

6. Устанавливается защищённое соединение, передаваемые данные шифруются.

Таким образом, SSL-сертификаты используются для обеспечения:

1. Конфиденциальности общения.
2. Подтверждения целостности сообщения (не было изменено в процессе доставки).
3. Подтверждения аутентичности источника сообщения.

VII. Программные средства для анализа безопасности Web-приложений

Web-приложение – это решение, в основе которого лежит взаимодействие браузера и web-сервера. Такие приложения являются кроссплатформенными сервисами, доступными с любого современного устройства, и не привязаны к архитектуре сети: получить к ним доступ можно с локального компьютера или со смартфона на другом конце света по наиболее удобному протоколу, например, более быстрому или зашифрованному. Сегодня Web-приложения составляют почти 95% информационных систем, поэтому необходимо тщательно подходить к их защите.

Тестирование web-приложений на проникновение – это метод оценки безопасности компьютерной системы или сети посредством методической проверки данных, а также проверки эффективности элементов управления безопасностью приложений. Проверка безопасности web-приложений фокусируется только на оценке безопасности web-приложения. Процесс включает в себя активный анализ приложения на наличие любого рода слабых сторон, технических недостатков или уязвимостей. Любые обнаруженные проблемы безопасности будут представлены владельцу системы вместе с оценкой воздействия, а также предложением по смягчению последствий данной проблемы или техническим её решением.

VII.1. Сканирование и поиск уязвимостей

Kali Linux

Для анализа и тестирования защищённости web-приложений и не только существует множество различных программ и методов. И одним из самых популярных на сегодняшний день является ОС Kali Linux, предустановленный функционал которой позволяет при правильном подходе совершить практически любой тип хакерской атаки, известной на данный момент. Изначально Kali

Linux была известна пользователям под несколько другим названием – BackTrack.

Основные преимущества этой операционной системы заключаются в следующем:

- более 300 инструментов для пентеста;
- свободное распространение;
- открытый исходный код;
- мультиязычность;
- безопасность;
- полностью настраиваемый (вплоть до ядра);
- мультиплатформенность.

Рис. VII.1. Предустановленное программное обеспечение для тестов web-приложений Kali Linux

И это далеко не весь список плюсов Kali. Но самым большим преимуществом, конечно же, является её инструментарий для различных задач, среди ко-

торых: сбор информации, анализ уязвимостей, беспроводные атаки, web-приложения, криминалистические инструменты, стресс-тестирование, инструменты эксплуатации, Sniffing и spoofing, атаки на пароли, поддержка доступа, обратная инженерия, взлом оборудования, шифрование данных и антикриминалистика, уязвимые среды и программы для тренировки, инструменты по составлению отчётов.

Обзор основных инструментов

1. Burp Suite.

Burp Suite – это интегрированная платформа для выполнения тестов по безопасности web-приложений. Её различные инструменты эффективно работают вместе для поддержки всего процесса тестирования, от составления карты сайта и анализа поверхности атаки приложения до поиска и эксплуатации уязвимостей безопасности.

Burp даёт полный контроль, позволяет комбинировать продвинутые ручные техники, что делает работу быстрее, более эффективной и более приятной.

Burp Suite содержит следующие ключевые компоненты:

- Перехватывающий прокси, который позволяет инспектировать и модифицировать траффик между вашим браузером и целевым приложением.
- Паук для приложений для обхода контента и функциональности.
- Продвинутый сканер web-приложений для автоматизированного выявления ряда типов уязвимостей.
- Инструмент Intruder для выполнения мощных пользовательских атак для поиска и эксплуатирования необычных уязвимостей.
- Инструмент Repeater для манипуляций и повторной отправки индивидуальных запросов.
- Инструмент Sequencer для тестирования хаотичных сессионных токенов (маркеров).

- Возможность сохранять работу и возобновлять рабочий процесс позже.
- Расширяемость, позволяющая вам легко писать ваши собственные плагины, для выполнения комплексных и высоко настраиваемых задач внутри Burp.

Burp прост и интуитивен, позволяет новичкам начать работу прямо сейчас. Burp также очень гибкий в настройке, содержит ряд мощных функций для помощи самым опытным тестером в их работе.

2. Commix.

Commix (это сокращение от [comm]and [i]njection e[x]ploiter) имеет простую среду и может использоваться web-разработчиками, тестерами на проникновение и даже исследователями безопасности для тестирования web-приложений в поисках багов, ошибок или уязвимостей, связанных с атаками инъекцией команд. С использованием этого инструмента очень легко находить и эксплуатировать уязвимости инъекции команд в конкретном уязвимом параметре или строке. Commix написан на языке программирования Python.

3. HTTrack.

HTTrack – это бесплатная, открытая, простая в использовании утилита для оффлайн просмотра web-сайтов.

Она позволяет загружать web-сайты из Интернета в локальный каталог, повторяя структуру директорий, получая HTML, изображения и другие файлы с сервера на компьютер. HTTrack организует оригинальную структуру относительных ссылок. Открыв страницу «зазеркаленного» web-сайта в браузере можно просматривать его, переходя по ссылкам так, будто бы мы просматриваем его онлайн. HTTrack также может обновить существующий зазеркаленный сайт и возобновить прерванные загрузки. HTTrack полностью настраиваемая утилита, имеет встроенную систему помощи.

4. Zaproxy.

OWASP Zed Attack Proxy (ZAP) — это один из самых популярных бесплатных инструментов безопасности, он активно поддерживается сотнями во-

лонтёров со всего мира. Zaproxy может помочь автоматически найти уязвимости безопасности в web-приложениях во время разработки и тестирования приложений. Это также отличный инструмент для опытных пентестеров для использования в ручном тестировании безопасности.

5. Sqlmap.

Sqlmap – это инструмент с открытым исходным кодом для тестирования на проникновение, который автоматизирует процесс выявления и эксплуатации уязвимости SQL-инъекция и захват серверов баз данных. Он поставляется с мощным движком выявления и многими нишевыми функциями для конечного тестера на проникновение, имеет широкий набор возможностей, начиная от сбора отпечатков баз данных по полученным от них данным, до доступа к файловой системе и выполнения команд в операционной системе посредством внеполосных (out-of-band) подключений.

6. WPScan.

WPScan – это сканер уязвимостей WordPress, работающий по принципу «чёрного ящика», т.е. без доступа к исходному коду. Он может быть использован для сканирования удалённых сайтов WordPress в поисках проблем безопасности.

Анализ уязвимостей в web-приложениях

Подбор ключа для замка

Узнать все необходимое о сайте, помогает программа whatweb. Благодаря этой программе можно получить информацию не только о титульной информации, находящейся на сайте, но также и получить сведения о версии движка сайта. А эта информация на порядок облегчает взломщику задачу, так как зная версию движка, можно быстрее найти необходимый эксплойт или известную уязвимость. И хотя скрыть версию движка зачастую не позволяет лицензия (и к тому же сделать это далеко не просто), многие web-разработчики идут на нарушение этой лицензии, лишь бы скрыть движок и его версию, тем самым обеспечить дополнительный уровень защиты данных.

Эксплойт – это программа, часть кода либо набор команд, который позволяет воспользоваться уязвимостями системы. Проще говоря, это ошибки в коде, используя которые, можно получить доступ как к частичному, так и к полному управлению конечной системой.

Чтобы воспользоваться программой, необходимо найти её в установленных программах в папке Web Apps. После этого в окне терминала необходимо будет ввести команду:

~# whatweb@ [адрес сайта]

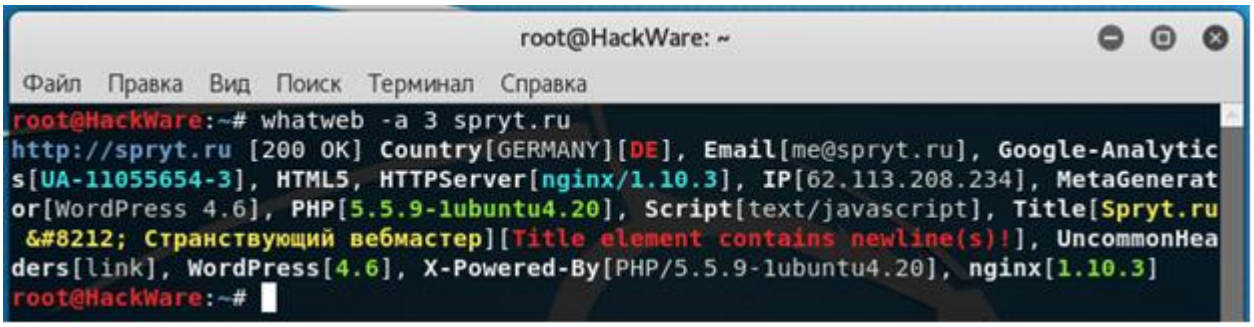


Рис. VII.2. Окно использования программы whatweb@[адрес сайта]

Получаем много полезной информации: страну расположения сайта, IP-адрес, версию движка и т.д. Теперь можно скачать исходники движка (в Сети для всех бесплатных) и поискать различные readme.txt, changelogs и т.д. Их, как правило не удаляют.

SQL-инъекции с помощью SQLMAP

Один из самых распространённых и доступных методов взлома web-страницы – это SQL-инъекции. Их суть заключается во внедрении в базу данных произвольного SQL-кода. Если сайт поддаётся, такой инъекции, то исходную БД можно менять как душе угодно, что в большинстве случаев используется либо для нарушения структуры, либо для полного удаления данных.

Программа SQLMAP позволяет тестировать ресурс на возможность такого проникновения. Благодаря мощному аналитическому движку и большому количеству специфичных функций, процесс выявления уязвимостей максимально автоматизирован. Также поддерживается шесть различных типов инъ-

екций, и полная поддержка систем управления БД: MySQL, Oralce, MS SQL Server, MS Access и т.д. Также есть возможность создания дампа таблиц, поддержка перебора пользователей, хешей паролей без данных, таблиц и колонок, а ещё есть возможность автоматического распознавания формата хеша пароля, и поддержка перебора пароля по словарю (брутфорс). И это далеко не весь список возможностей этой программы.

Прежде чем выявить уязвимый сайт необходимо сделать правильный запрос с помощью Google Dorks. Это специальный набор запросов для выявления пробелов в безопасности. Выглядеть они могут следующим образом:

```
inurl:item_id=
inurl:review.php?id=
inurl:hosting_info.php?id=
inurl:newsid=
```

Здесь используется оператор inurl, который указывает на то, что искомое слово должно быть частью страницы или сайта. Ещё есть операторы site для того, чтобы искать по конкретному сайту, intitle для поиска в заголовке страниц и ext (или filetype) для поиска файлов по расширению.

Metasploit как инструмент для поиска уязвимостей

Ещё одной крайне полезной программой является Metasploit Framework. Это огромный массив подпрограмм, позволяющий искать информацию об уязвимостях, создавать сигнатуры для IDS, а также создавать и тестировать эксплойты.

Всё более актуальной становится проблема защиты информации. Соответственно растёт спрос на профессионалов пентеста (теста на проникновение). Современный специалист должен не только досконально знать систему, с которой работает, но и средства, с помощью которых он может эту систему защитить. И ОС Kali Linux на сегодняшний день является фаворитом среди бесплатных инструментов. Эта система обладает мощным функционалом, в чем мы убедились выше. Разумеется, пользователям также необходимо повышать свою киберграмотность. Тогда будет возможно создание симбиоза пользователь-

разработчик, который сможет создавать все более защищённые системы и web-приложения.

Acunetix

Протокол HTTP может использовать множество способов кодирования и инкапсуляции информации, поэтому контроль за вводимой информацией и проверка её корректности по-прежнему являются ключевыми факторами при обеспечении безопасности всех web-приложений.

Вручную проверить большой объем кода, чтобы выявить все возможные значения всех элементов данных – задача сложная и трудоёмкая, требующая больших знаний в различных областях. По этой причине все чаще приходится прибегать к специализированным сканерам безопасности, работающим на прикладном уровне. Одним из продуктов, позволяющим протестировать web-сайт на наличие слабых мест, является Acunetix Web Vulnerability Scanner.

Acunetix Web Vulnerability Scanner позволяет быстро и эффективно защитить web-сайты и web-приложения, упрощая обнаружение уязвимостей, в сценариях на языках PHP, ASP, ASP NET. Помимо этого, программа может проверять на ошибки сценарии на JavaScript. Тесты можно проводить не только на прямую, но и через прокси-сервер для обеспечения анонимности. Также производит тестирование аутентификации HTTP для определения небезопасных настроек и ненадёжных паролей.

Сканер определяет хосты, на который работают службы сервера, и сканирует данные хосты на присутствие общих слабых мест, таких как перекрёстные коды, просмотр-обход директорий, ввод SQL и многих других. Программа также производит тестирование аутентификации HTTP для определения небезопасных настоек и ненадёжных. Она сообщает о наличии устаревшего программного обеспечения на сервере, которое необходимо обновить или исправить.

С её помощью можно просканировать виртуальные хосты и все типы web-серверов, а не только на основе Windows. WVS поддерживает ОС UNIX и

Linux, web-службы Apache и файлы настроек PHP. Имеется возможность подвергнуть ОС специальным проверкам.

Можно просканировать множество типов web-страниц и файлов, включая ASP и CGI. Программа поддерживает безопасные прокси и HTTP (HTTPS) и SOCKS сервера.

Acunetix имеет достаточно простой в использовании интерфейс. После входа в систему пользователи попадают на панель инструментов, которая даёт представление о текущем состоянии безопасности ресурсов сайта.

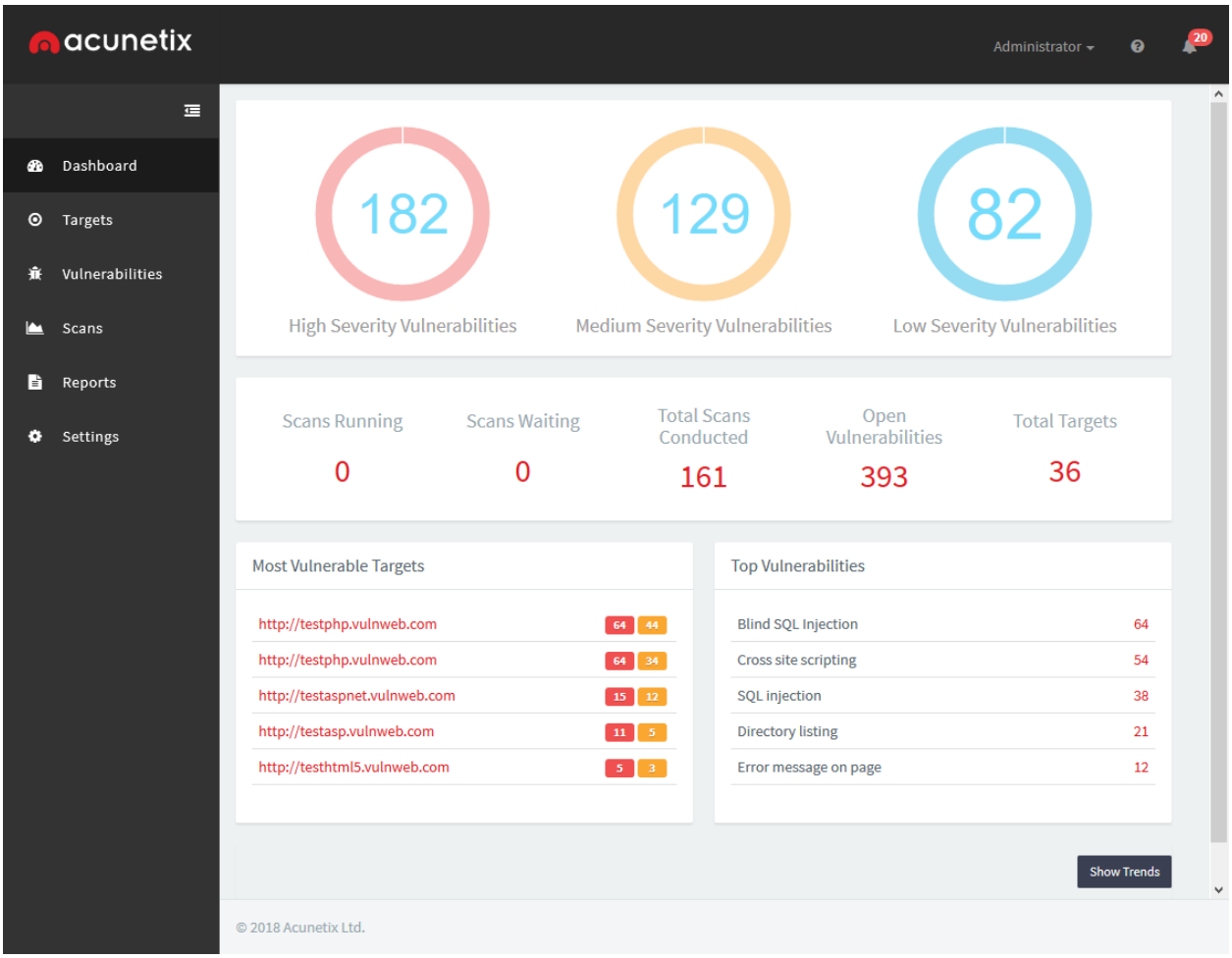


Рис. VII.3. Окно программы Acunetix

Отсюда пользователи могут получить доступ к встроенным функциям управления уязвимостями. У программы существуют свои преимущества, к ним относят:

- Удобная настройка целей сканирования – при однократном настраивании цели в дальнейшем её можно использовать постоянно без повторной настройки. Acunetix отслеживает состояние безопасности каждой цели путём агрегирования и отслеживания уязвимостей, определённых для каждой из них.
- Все уязвимости, выявленные Acunetix, показаны на одной странице, что упрощает расстановку приоритетов. Уязвимости могут быть отфильтрованы или сгруппированы.
- Возможность просматривать результаты сканирования текущих или завершённых проверок. Сканирование можно осуществить мгновенно или установить расписание проверок.

Программа использует проверку безопасности сайта, состоящую из двух этапов:

1. Используя Acunetix DeepScan, программа автоматически анализирует и сканирует web-сайт для построения его структуры. Процесс сканирования перечисляет все файлы, папки и входные данные.
2. Затем запускается серия проверок web-уязвимостей для каждого компонента в вашем web-приложении – фактически эмулируя хакера. Результаты сканирования включают подробные сведения обо всех уязвимостях, обнаруженных на web-сайте.

Программа включает в себя уникальную технологию Acunetix AcuSensor Technology, которая позволяет выявлять больше уязвимостей, чем традиционные web-сканеры безопасности. Кроме того, он также указывает строку кода, в которой была обнаружена уязвимость. Для работы Acunetix должен быть установлен агент, обеспечивающий связь между Acunetix и AcuSensor. Acunetix AcuSensor можно использовать с web-приложениями на PHP, JAVA и .NET.

Программа позволяет создавать отчёты для сканирования, целей и всех обнаруженных уязвимостей. Доступны различные шаблоны отчётов, включая резюме, подробные отчёты и широкий спектр отчётов о соответствии.

VII.2. Проведение нагрузочных тестов

Web-приложения массового обслуживания пользователей для определения надёжности и отказоустойчивости проверяют с помощью нагрузочного тестирования. Существуют различные решения для проведения нагрузочного тестирования web-приложений.

Одним из таких инструментов, для проведения таких тестов, считается Apache JMetr.

Apache JMetr

Apache JMeter – специальная программа, инструмент, который используется для проведения нагрузочного тестирования. JMeter имеет возможности по созданию нагрузочного тестирования с большим количеством настраиваемых параметров, также у него имеется поддержка дополнительных плагинов, с использованием которым функционал программы значительно расширяется. В рамках выполнения курсового проекта подойдет и стандартная версия программы, настраиваемых параметров будет достаточно для создания различных нагрузочных тестов для проверки эффективности и безопасности плагинов электронной коммерции.

Для скачивания программы переходим на сайт «jmeter.apache.org», где нам откроется страничка «Download Apache JMeter». В настоящий момент свежая версия данной программы – 5.11, которая требует обязательной установки Java версии 8 и выше (поэтому, перед тем как скачивать и устанавливать Apache JMeter необходимо скачать Java версии 8 и выше, иначе поработать с JMeter не получится). На странице «Download Apache JMeter» имеются два списка – «Binaries» и «Source». Нас интересует список «Binaries» и файл «apache-jmeter-5.1.1.zip» - ссылка на скачивание архива. Нажимаем на название и скачиваем архив с программой «Apache JMeter 5.1.1». После этого создаем какую-либо папку, куда распаковываем скачанный архив (важно – путь в этой папке не должен содержать пробелы и кириллицу). В моем случае эта папка располагается по пути «C:\Users\User\Desktop» и называется она «tools». Распа-

ковываем скачанный архив в эту папку и открываем её, дабы ознакомится с содержимым (рис. VII.4).

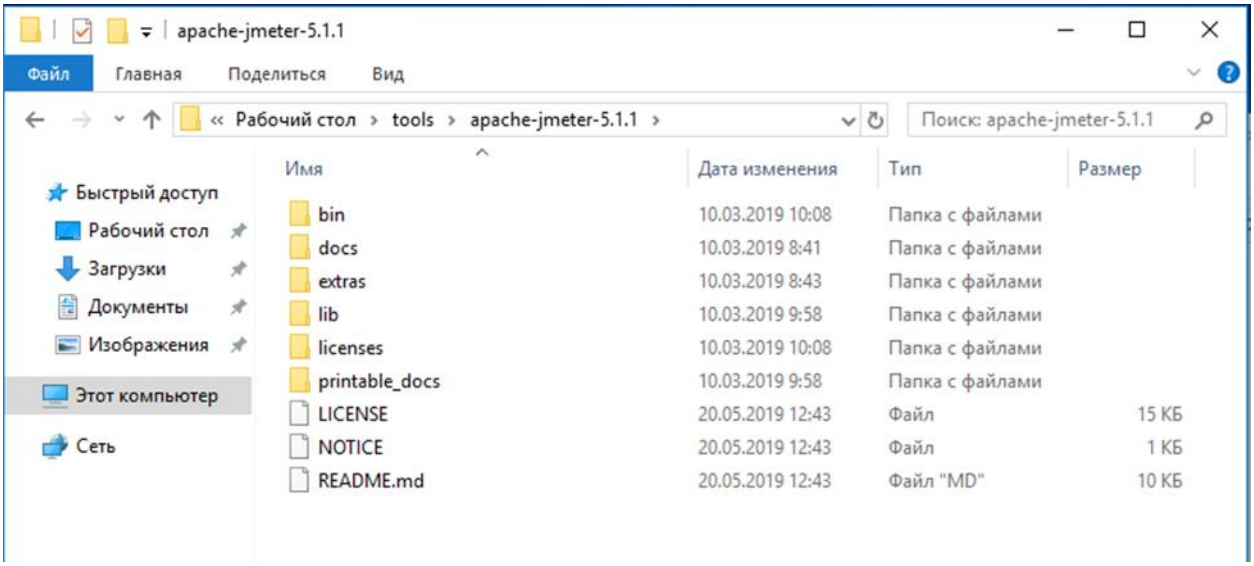


Рис. VII.4. Файлы Apache JMeter

На самом деле нам необязательно исследовать каждый каталог, в рамках курсовой работы ценность представляет только папка «bin», так как в ней находятся исполняемые файлы программы, однако, считаю нужным обозначить назначение каталогов и файлов:

- Папка bin хранит в себе исполняемые файлы, например, «jmeter.bat»;
- Папка docs содержит документацию по программе;
- Папка extras содержит некоторые дополнительные скрипты и расширения;
- Папка lib хранит используемые библиотеки;
- Папка licenses – лицензии;
- Папка printable_docs, исходя из названия, также содержит документацию;
- Файл LICENSE – лицензия Apache 2.0 на Apache.JMeter;
- Файл NOTICE – примечания;
- Файл README.md – описание, требования и инструкция по установке и сборке.

Как уже говорилось выше, для выполнения курсовой работы необходимо перейти в папку «bin», найти там файл «jmeter.bat» и запустить его (рис. VII.5.). Это и есть сама программа Apache JMeter, которую мы будем использовать для тестирования плагинов электронной коммерции.

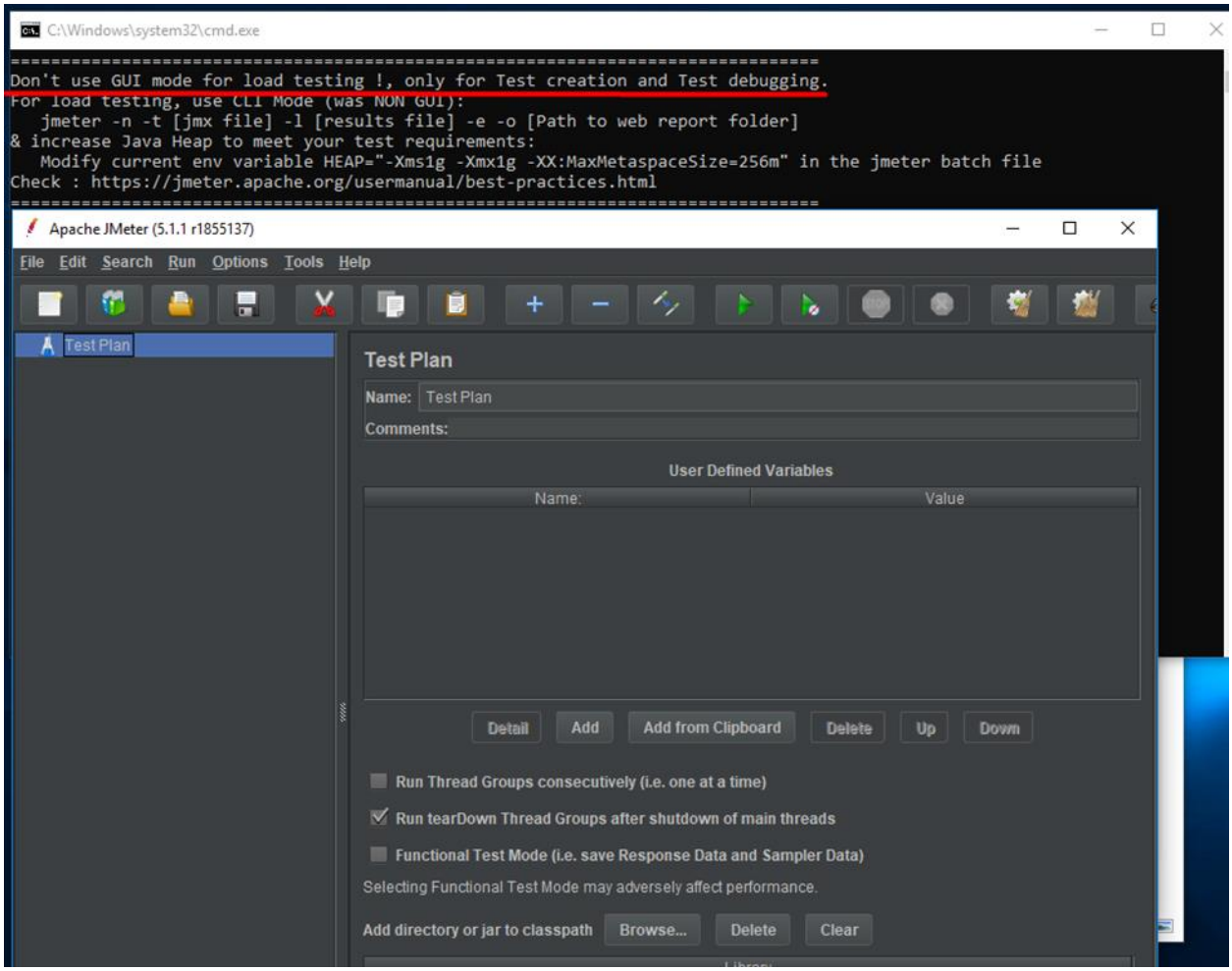


Рис. VII.5. Программа Apache JMeter готова к созданию и настройке нагрузочных тестов

Обратите внимание на строчку, подчеркнутую красным (рис. VII.5.), оказывается, не рекомендуется использовать GUI при проведении нагрузочных тестов, его следует использовать только при создании тестов и их отладке.

Список используемой литературы

1. Acunetix | Web Application Security Scanner. [Электронный ресурс] – Режим доступа: https://www.acunetix.com/ - Дата доступа: 28.04.2020.
2. Centralized Web Farm Management. Using the FTP Service. [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/iis/publish/using-the-ftp-service/ftp-extension-video-walkthrough - Дата доступа: 21.04.2020.
3. Cisco Systems, Inc., которая является мировым лидером на рынке сетевых решений для Internet [Электронный ресурс] – Режим доступа: https://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-/index.html/index.html - Дата доступа: 02.02.2020
4. Configuring FTP Firewall Support. [Электронный ресурс] – Режим доступа: http://technet.microsoft.com/ru-ru/library/dd464003(WS.10).aspx - Дата доступа: 21.04.2020.
5. DDoS-атаки: типы атак и уровни модели OSI [Электронный ресурс] – Режим доступа: https://firstvds.ru/technology/types-of-ddos - Дата доступа: 12.04.2020.
6. Faircloth J. Penetration Tester's Open Source Toolkit / J. Faircloth. Elsevier, 2011. – 464 p.
7. Installing and Configuring FTP on IIS 7. [Электронный ресурс] – Режим доступа: http://learn.iis.net/page.aspx/263/installing-and-configuring-ftp-on-iis-7/ - Дата доступа: 21.04.2020.
8. ModSecurity: Open Source Web Application Firewall [Электронный ресурс] – Режим доступа: http://modsecurity.org - Дата доступа: 21.03.2020.
9. OWASP ModSecurity Core Rule Set [Электронный ресурс] – Режим доступа: https://owasp.org/www-project-modsecurity-core-rule-set/ - Дата доступа: 21.03.2020.

10. OWASP Risk Rating Methodology [Электронный ресурс] – Режим доступа: https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology - Дата доступа: 02.05.2020.

11. OWASP Top Ten Project [Электронный ресурс] – Режим доступа: https://www.owasp.org/index.php/Top10 - Дата доступа: 02.05.2020.

12. PuTTY: Telnet/SSH Клиент. [Электронный ресурс] – Режим доступа: http://putty.org.ru - Дата доступа: 21.04.2020.

13. Rosner G. Privacy and the Internet of things. [Электронный ресурс] – Режим доступа: https://www.goodreads.com/book/show/32804476-privacy-and-the-internet-of-things - Дата доступа: 28.02.2020.

14. Secure Sockets Layer [Электронный ресурс] – Режим доступа: http://home.netscape.com/eng/ssl3/index.html - Дата доступа: 21.03.2020.

15. SQL инъекции. Проверка, взлом, защита [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/130826/ - Дата доступа: 12.04.2020.

16. The Open Web Application Security Project [Электронный ресурс] – Режим доступа: https://www.owasp.org/index.php/Main_Page - Дата доступа: 02.05.2020.

17. Web Application Firewall (ModSecurity) в действии [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/228103/ - Дата доступа: 21.03.2020.

18. Адамов А.В., Бабич А.В. Универсальный метод защиты веб-приложений. [Электронный ресурс] – Режим доступа: http://conf.nsc.ru/files/conferences/ym2014/fulltext/247333 - Дата доступа: 21.03.2020.

19. Барабанова М.И., Кияев В.И. Информационные технологии: открытые системы, сети, безопасность в системах и сетях / М.И. Барабанова, В.И. Кияев. Учебное пособие. – СПб. : Изд-во СПбГУЭФ, 2010. – 267 с.

20. Бондарев В.В. Введение в информационную безопасность автоматизированных систем. / В.В. Бондарев. Учебник. – М. : МГТУ им. Н. Э. Баумана, 2016. – 252 с.

21. Калинин Н., Шерварлы В., Петухов А. Общий обзор классификаций угроз безопасности: OWASP, CWE, CAPEC, WASC [Электронный ресурс] – Режим доступа: https://safe-surf.ru/specialists/article/5210/595970/?sphrase_id=45471 – Дата доступа: 02.05.2020.

22. Леонтьев Б.К. Хакинг без секретов. / Б.К. Леонтьев. Справочное руководство пользователя ПК. – М. : Познавательная книга плюс, 2000. – 734 с. : ил.

23. Медведовский И.Д., Семьянов П.В., Леонов Д.Г. Атака на Internet. / И.Д. Медведовский, П.В. Семьянов, Д.Г. Леонов М. - 2-е изд., перераб. и доп. – М. : ДМК Пресс, 2006. – 336 с. : ил.

24. Петренко С.А., Курбатов В.А. Политики информационной безопасности / С.А. Петренко, В.А. Курбатов. Информационные технологии для инженеров. – М. : Компания АйТи, 2010. – 400 с. : ил.

25. Разработка системы управления содержимым сайта. Классификация web-приложений. [Электронный ресурс] – Режим доступа: https://studbooks.net/2247982/informatika/klassifikatsiya_prilozheniy - Дата доступа: 28.04.2020.

26. Ристик И., перевод Антипов А. Защита Web приложений с помощью Apache и mod_security. [Электронный ресурс] – Режим доступа: https://www.securitylab.ru/analytics/216322.php - Дата доступа: 21.03.2020.

27. Сычев, А.В. Web-технологии / А.В. Сычев. – М. : НОУ «Интуит», 2016. – 408 с.

28. Уроки по XSS: Урок 1. Основы XSS и поиск уязвимых к XSS сайтов [Электронный ресурс] – Режим доступа: https://hackware.ru/?p=1174 - Дата доступа: 12.04.2020.

29. Ховард М., Лебланк Д., Вьега Дж. 24 смертных греха компьютерной безопасности. Библиотека программиста. / – СПб.: Питер, 2010. – 400 с.: ил.

30. Цифровой сертификат безопасности: для чего это нужно? [Электронный ресурс] – Режим доступа: https://habr.com/ru/company/regru/blog/280878/ - Дата доступа: 21.03.2020.