Министерство образования и науки Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
—
Институт компьютерных наук и технологий
Кафедра «Информационная безопасность компьютерных систем»

Работа допущена к защите
Заведующий кафедрой
д.т.н., проф.
_____ П.Д. Зегжда
"____"_____ 2015 г.

ДИПЛОМНАЯ РАБОТА СПЕЦИАЛИСТА

МОДЕЛИРОВАНИЕ СЕТЕВЫХ АТАК ДЛЯ ТЕСТИРОВАНИЯ СРЕДСТВ ЗАЩИТЫ ИНФОРМАЦИИ

по специальности 090105.65 – Комплексное обеспечение информационной безопасности автоматизированных систем

Выполнил
студент гр. 53508/ 3 А.М. Смирнов

Научный руководитель
профессор, к.т.н. В.В. Платонов

Санкт-Петербург
2015

ФГАОУ ВО «Санкт-Петербургский политехнический университет Петра Великого»
Институт информационных технологий и управления
Кафедра «Информационная безопасность компьютерных систем»

УТВЕРЖДАЮ
Зав. кафедрой
д.т.н., проф.
_____П.Д. Зегжда
«____» _____ 2015 г.

ЗАДАНИЕ

на дипломную работу

студенту _____группы 53508/3___

1. Тема работы_____
_____

2. Срок сдачи студентом законченной работы_____

3. Исходные данные к работе_____
_____
_____
_____
_____
_____
_____

4. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов) _____
_____
_____
_____
_____
_____

5. Дата выдачи задания_____

Руководитель _____ (_____)
(подпись руководителя) (Фамилия и инициалы)

Задание принял к исполнению «____» _____ 2015 г.
_____ (_____)
(подпись студента) (Фамилия и инициалы)

РЕФЕРАТ

Работа состоит из: 59 с., 15 рис., 4 табл., 39 источников.

МОДЕЛИРОВАНИЕ СЕТЕВЫХ АТАК, АНАЛИЗ ЗАЩИЩЕННОСТИ СИСТЕМЫ, ТЕСТИРОВАНИЕ СРЕДСТВ ЗАЩИТЫ ИНФОРМАЦИИ, АВТМАТИЗИРОВАННЫЙ ЭКСПЛОЙТИНГ, ТЕСТИРОВАНИЕ НА ПРОНИКНОВЕНИЕ

В настоящей дипломной работе произведен анализ методов тестирования средств защиты информации, разработан собственный метод и реализован прототип программного средства, позволяющий автоматизировать проведение тестов на проникновение. Полученное средство позволяет эффективно тестировать защищенность компьютерных систем за счет автоматического выбора подходящих под конкретную цель эксплойтов и возможности использовать собственные, сторонние эксплойты.

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ 4
ВВЕДЕНИЕ 6
1 Проблема сетевых атак 8
1.1 Уязвимости 9
1.2 Эксплойты 14
1.2.1 Принципы работы эксплойтов 14
1.2.2 Классификация эксплойтов 17
1.2.3 Виды эксплойтов 17
2 Методы тестирования средств защиты информации 19
2.1 Armitage 21
3 Архитектура системы 23
3.1 Подсистема сбора информации 24
3.2 Подсистема формирования БД эксплойтов 31
3.3 База данных анализатора 32
3.4 Metasploit Framework 32
3.5 Подсистема анализа защищенности 33
4 Разработка программного комплекса 38
4.1 Metasploit 38
4.1.1 Ранжирование эксплойтов в Metasploit. 41
4.1.2 Типы модулей в MSF 42
4.2 Подсистема формирования БД эксплойтов 44
4.3 Подсистема сбора информации 46
4.4 Подсистема анализа защищенности 49
5 Тестирование разработанного программного комплекса 52
ЗАКЛЮЧЕНИЕ 55
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 56

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

АС	Автоматизированная система
БД	База данных
ВПО	Вредоносное программное обеспечение
ИТ	Информационные технологии
МЭ	Межсетевой экран
ОС	Операционная система
ПО	Программное обеспечение
СОВ	Система обнаружения вторжений
СПВ	Система предотвращения вторжений
СУБД	Система управления базой данных
CSRF	Cross Site Request Forgery – Межсайтовая подделка запроса
DLL	Dynamic Link Library – Динамически подключаемая библиотека
GUI	Graphical User Interface – Графический пользовательский интерфейс
LFI	Local File Inclusion – Подключение локальных файлов
MP	Meterpreter – Расширенная многофункциональная полезная нагрузка
MSF	Metasploit Framework – Инструмент для создания, тестирования и использования эксплойтов
MTU	Maximum transmission unit – Максимальный размер полезного блока данных одного пакета
RFI	Remote File Inclusion – Подключение удаленных файлов
SAM	Security Account Manager – Диспетчер учётных записей безопасности
XSS	Cross Site Scripting – Межсайтовый скриптинг

ВВЕДЕНИЕ

На протяжении нескольких последних десятилетий информационные технологии (ИТ) все больше и больше входят в нашу жизнь. А вместе с ростом вовлеченности ИТ в различные сферы человеческой деятельности растет и значимость информационной безопасности, так как появляется множество незаконных способов получения доступа к информации, расположенной на удаленных хостах.

Нарушение информационной безопасности может быть вызвано рядом причин: уязвимости в установленных приложениях или самой операционной системе (ОС), неправильная конфигурация систем обнаружения вторжений (СОВ), антивирусов и других защитных средств, ошибки при настройке системы разграничения доступа, просчеты в самой политике безопасности и так далее.

Вследствие наличия подобных нарушений безопасности в автоматизированных системах (АС), злоумышленник может, в зависимости от своих целей, осуществить один из сценариев атак, таких как получение доступа к системе, повышение привилегий пользователя, внедрение вредоносного программного обеспечения и других.

Поэтому, на любом этапе жизненного цикла АС, будь то проектирование, конфигурирование или эксплуатация, администратору необходимо знать, действительно ли система удовлетворяет требуемому уровню защищенности. На этапе конфигурирования и настройки АС требуется проверить, правильно ли подобраны и настроены средства защиты информации, выбрана ли верная политика безопасности и так далее. Во время эксплуатации системы также следует регулярно осуществлять аудит безопасности, так как любая система со временем меняется, добавляются новые компоненты, а, следовательно, появляются новые угрозы.

С ростом сложности компьютерных систем и механизмов защиты, увеличением числа известных уязвимостей и ошибок в действиях пользователей, появляется острая необходимость в автоматизации процесса анализа защищенности АС и выявления содержащихся в них уязвимостей, так как ручная проверка

занимает огромное количество времени и требует от аудитора многократного повторения однотипных действий.

Существующие автоматические средства проверки на наличие уязвимостей, такие как auto_pwn в Metasploit Framework, основаны на последовательном запуске большого множества эксплойтов, что является не слишком эффективным и довольно грубым методом.

В данной работе представлен улучшенный метод автоматического подбора и запуска эксплойтов, основанный на проведении предварительной разведки с целью определения версии ОС, пакета обновлений, языка целевой системы, списка открытых портов и запущенных сервисов, а после – интеллектуального выбора эксплойтов по этим данным. База данных эксплойтов формируется путем категоризации и группировки эксплойтов из общедоступных списков. В результате, разработанный программный комплекс позволяет оценивать уровень защищенности, предоставляемый тем или иным средством защиты информации, и показывает список уязвимостей, которые удалось успешно эксплуатировать.

1 Проблема сетевых атак

По данным Лаборатории Касперского [1] за 2014 год было заблокировано более 6000000000 вредоносных атак, направленных на компьютеры и мобильные устройства. Атаки были проведены с 9 000 000 уникальных хостов. 44% веб-атак проводились с использованием вредоносных веб-ресурсов, расположенных в США и Германии. Таким образом, в течение одного года 38,3% компьютеров, подключенных к интернету, хотя бы подвергались веб-атаке. Антивирусом «Kaspersky Antivirus» было обнаружено более 120 000 000 уникальных вредоносных программ (скриптов, эксплойтов, исполняемых файлов).

Одним из способов проникновения в компьютерные системы является эксплуатация уязвимостей (рисунок 1). Этот способ эффективен благодаря наличию уязвимостей в широко используемом программном обеспечении. Кроме того, отдельные пользователи и компании не устанавливают дополнения и исправления, закрывающие известные уязвимости в приложениях.

Рисунок 1 – Рост числа уязвимостей за 2008–2014 года

1.1 Уязвимости

Введем определение некоторых понятий, которые будем в дальнейшем использовать. Уязвимость информационной системы – это свойство информационной системы, обусловливающее возможность реализации угроз безопасности обрабатываемой в ней информации [2]. Другими словами, уязвимость – это ошибка или упущение в исходном коде программы, которые злоумышленник может использовать в своих целях.

Существует множество различных классификаций уязвимостей: по степени критичности [3], источнику возникновения, уровню в инфраструктуре АС [4] и так далее. Ниже представлена классификация уязвимостей на основе причин их возникновения (таблица 1) [5].

Таблица 1 Классификация уязвимостей по причинам их возникновения

Тип	Класс	Группа	Вид
Тип 1. Уязвимости, вызванные дефектами кодирования и проектирования системы	Класс 1. Обработка и представление данных	Группа 1.1. Обработка входных и выходных данных	Вид 1.1.1. Проверка и представление ввода
			Вид 1.1.2. Некорректное кодирование и экранирование вывода
			Вид 1.1.3. Некорректная обработка синтаксически

			неверных структур
		Группа 1.2. Внутренние трансформации данных	Вид 1.2.1. Ошибки строк
			Вид 1.2.2. Ошибки типов
			Вид 1.2.3. Ошибки представления
			Вид 1.2.4. Числовые ошибки
			Вид 1.2.5. Проблемы структур данных
		Группа 1.3. Ошибки доступа к данным	Вид 1.3.1. Ошибки управления информацией
			Вид 1.3.2. Неверный доступ к индексируемому ресурсу
			Вид 1.3.3. Модификация постоянных данных
	Класс 2. Внутренняя структура и зависимости	Группа 2.1. Злоупотребление API	
		Группа 2.2. Инкапсуляция	
	Класс 3.	Группа 3.1.	

	Обработка событий и состояний	Время и внутреннее состояние	
		Группа 3.2. Проблемы с логикой функционирования	
		Группа 3.3. Проблемы с обработчиками	Вид 3.3.1. Обработка ошибок и внештатных ситуаций
	Класс 4. Ресурсы и внутренние механизмы системы	Группа 4.1. Механизмы безопасности	
		Группа 4.2. Ошибки каналов и путей	
		Группа 4.3. Ошибки инициализации и очистки	
		Группа 4.4. Проблемы ссылок и псевдонимов	Вид 4.4.1. Проблемы с указателями
		Группа 4.5. Ошибки свойственные определенному типу функционала	Вид 4.5.1. Пользовательский интерфейс
			Вид 4.5.2. Проблемы WEB

	Класс 5. Внедренные объекты (закладки)	Группа 5.1. Намеренное внедренные объекты	
		Группа 5.2. Внедренные ненамеренно объекты	
	Класс 6. Качество проектирования, реализации, документирования	Группа 6.1. Качество кода	
		Группа 6.2. Нарушение принципов проектирования безопасного ПО	
		Группа 6.3. Неполная или некорректная документация	
Тип 2. Уязвимости, вызванные дефектами конфигурирования и управления системой и ее окружением	Класс 7. Конфигурация	Группа 7.1. Настройки механизмов безопасности	
		Группа 7.2. Настройки структуры и функционала	
		Группа 7.3. Закладки в	

		настройках	
		Группа 7.4. Совместимость версий	
		Группа 7.5. Качество настроек	
	Класс 8. Окружение	Группа 8.1. Среда компиляции и выполнения программного кода	
		Группа 8.2. Прикладное программное обеспечение	
		Группа 8.3. Системное программное обеспечение (гипервизор, ОС, драйвера)	
		Группа 8.4. Аппаратное обеспечение	

Наиболее полный и структурированный список уязвимостей имеет название Common Vulnerabilities and Exposures (CVE) [6]. В нем содержатся перечень всех общеизвестных уязвимостей. Основное преимущество CVE в том, что все уязвимости в нем систематизированы, каждая имеет свой номер и описание. В свою очередь большинство эксплойтов ссылаются именно на CVE, чтобы можно было легко определить, какую уязвимость использует тот или иной эксплойт.

1.2 Эксплойты

Эксплойт – это программа, последовательность команд или часть программного кода, использующие уязвимости в ПО для проведения атаки на АС. Цели атаки могут быть самыми разнообразными: получение доступа, нарушение стандартной функциональности системы и прочее.

Эксплойты могут представлять собой исходный код, исполняемый модуль или даже словесное описание того, как надо использовать уязвимость. Он может быть написан практически на любом языке программирования: C/C++, Perl, Ruby, JavaScript, Assembler.

1.2.1 Принципы работы эксплойтов

Чтобы понять, как работают эксплойты, необходимо знать основные принципы организации памяти [7]. Процесс, находящийся в памяти, имеет следующие подструктуры:

- сегмент кода – содержит скомпилированный исполняемый код программы;
- сегмент данных – содержит статические, глобальные, инициализированные и неинициализированные данные и переменные;
- стек – структура данных, работающая по принципы FIFO (First In First Out). Элементы помещаются в вершину стека и удаляются также с вершины. Регистр SP (Stack Pointer) содержит указатель на вершину стека. Запускаемый процесс помещает все свои данные в стек. Регистр IP содержит адрес команды, которую следующей выполнит процессор. Если имеет место резкое изменение места выполнения программы (обычно, вследствие jmp или call), после возвращения назад, адрес следующей команды может быть потерян. Для решения этой проблемы, программа хранит адрес следующей команды, которая должна быть выполнена (после возврата из jmp или call), в стеке. Этот адрес называется адрес возврата

(реализован в ассемблерной команде RET). Это обеспечивает правильное выполнение программы, содержащей много вызовов функций и goto команд [8];

- куча – остальная память, выделенная процессу.

Далее рассмотрим основные методы атак, использующие особенности строения памяти процессов.

Считается, что переполнение буфера – одна из самых опасных угроз безопасности за последнее десятилетие [9]. Стек программы хранит данные в следующем порядке: параметры, которые были переданы функции, затем адрес возврата, далее предыдущее значение указателя стека и локальные переменные. Если переменные (например, массивы) передаются без проверки границ, то при большом количестве данных, они могут разрушить стек, переписав адрес возврата и, следовательно, вызвав ошибку сегментации. Если правильно реализовать эту уязвимость, то можно изменить буфер так, чтобы он указывал на любой адрес, например тот, где лежит вредоносный код [10].

Помимо переполнения буфера возможно еще переполнение кучи. Куча обычно представлена как двусвязный список. При переполнении кучи можно изменить указатели таким образом, что они будут указывать на участок кучи со зловредным кодом. Переполнения кучи трудно вызвать и они более характерны для ОС Windows, так она содержит данные, которые могут использоваться для создания эксплойта. В случае системы распределения памяти функцией malloc, информация о свободной и распределенной памяти хранится в пределах кучи. Переполнение может быть вызвано использованием этой информации таким образом, чтобы впоследствии можно было писать в случайные области памяти, что может привести к исполнению кода нарушителя [11].

Для того чтобы вызвать переполнение, есть много различных способов, например строки и строковые функции, строки формата, нулевые указатели, целочисленные переполнения, известные проблемы и ситуации, который могут помочь создать исключительную ситуацию для процесса [12].

Ниже представлены основные составляющие эксплойта (рисунок 2).

Рисунок 2 – Составляющие эксплойта

Шеллкод – это полезная нагрузка, которая выполняется после успешного запуска эксплойта. Чаще всего адрес возврата меняется на адрес памяти, где расположен этот шеллкод [13**Ошибка! Источник ссылки не найден.**]. Шеллкод представляет собой ассемблерные команды, закодированные в виде двоичной строки. При написании шеллкода необходимо соблюдать баланс между сложностью и размером кода, также есть определенные ограничения, накладываемые на шеллкода (например, отсутствие некоторых символов). С помощью шеллкода можно выполнять различные операции, такие как открытие сокета, загрузка ВПО, запуск командной строки и так далее.

Вектор инъекции – это указатель или смещение, по которому внедряется шеллкод и измененный адрес возврата, указывающий на это место в памяти.

Построитель запроса – это код, который вызывает эксплойт.

Стандартный обработчик – это обработчик шеллкода, который выполняет такие операции, как связь консоли с сокетом или создание биндшелла (соединение с атакуемой машиной типа «клиент-сервер»).

Обработчик пользовательских опций – это пользовательский интерфейс, который предоставляет пользователю выбирать некоторые опции, такие как выбор удаленной цели, размер смещения, дополнительная информации, отладка и тому подобное.

Обработчик сетевых соединений – различные утилиты, упрощающие работу с сетью, например определение IP-адреса по имени хоста, установка соединений, обработка ошибок.

1.2.2 Классификация эксплойтов

В зависимости от способа получения доступа к уязвимой системе, различают локальные и удаленные эксплойты. Для запуска локального эксплойта необходимо иметь доступ к целевому хосту. Обычно используется для повышения прав пользователя. Удаленный же эксплойт можно запустить через сеть, не имея прямого доступа к системе.

По типу используемой уязвимости эксплойты можно разделить на эксплойты, направленные на: переполнение буфера, SQL инъекции, XSS и CSRF атаки и так далее.

1.2.3 Виды эксплойтов.

Принято разделять эксплойты по видам целей, для которых они предназначены:

1. Эксплойты для операционных систем.
2. Эксплойты для прикладного ПО (музыкальные проигрыватели, офисные пакеты и т. д.).
3. Эксплойты для браузеров (Chrome, Internet Explorer, Mozilla Firefox, Opera и другие).
4. Эксплойты для интернет-приложений, фреймворков (IPB, WordPress, VBulletin, phpBB).
5. Эксплойты для интернет-сайтов (facebook.com, hi5.com, livejournal.com).
6. Другие эксплойты.

Как только новая уязвимость появляется в открытом доступе, ей могут воспользоваться как нарушители для проведения атак, так и производители ПО для закрытия этой уязвимости. После закрытия уязвимости все эксплойты,

использовавшие ее, скорее всего, станут бесполезны, поэтому наиболее популярными и востребованными являются так называемые «zero day» эксплойты, построенные на только что открытых уязвимостях. Вероятность их срабатывания очень высока.

Для блокировки эксплойтов применяются всевозможные средства защиты, такие как: антивирусы, межсетевые экраны, системы обнаружения и предотвращения вторжений и другие. Однако, для корректной работы этих программ необходима сложная и детальная настройка. Администратор хоста не может сразу определить, достаточно ли защищена его система, все ли уязвимости исправлены, правильно ли сконфигурированы программы, поэтому необходимо проводить аудит безопасности, который, в том числе, включает в себя тестирование средств защиты информации.

2 Методы тестирования средств защиты информации

Существует несколько подходов к тестированию безопасности системы. Во-первых, можно провести ряд тестов на проникновение (пентестов), то есть, используя всевозможные утилиты, базы данных уязвимостей и эксплойтов попытаться получить доступ к целевой системе, серверу, компьютеру, обойти тестируемую программу. Преимущество пентестов в том, что их проводят люди, соответственно, они могут в реальном времени принимать те или иные решения, менять стратегии, средства атак и прочее. Однако, в этом же и недостаток данного подхода – все зависит от компетентности тех, кто эти пентесты проводит. Нельзя после проведения ряда атак утверждать, что система обладает нужным уровнем защищенности. Ведь возможно, что просто были выбраны не те средства или использовались устаревшие уязвимости.

Второй метод заключается в воспроизведении предварительно записанного трафика компьютерных атак. Настраивается специальный стенд, к которому подключается тестируемая система. Преимуществом этого подхода является возможность обеспечения многократной повторяемости условий эксперимента при условии, что трафик компьютерных атак каждый раз будет воспроизводиться идентичным образом. Отсутствие в составе стенда реальных атакуемых систем устраняет проблему восстановления состояния отдельных его элементов после проведения атак. Но рассматриваемый способ тестирования не позволяет варьировать параметры атак (например, кодировки HTTP-запросов и используемый эксплойтом shell-код) в процессе тестирования, поскольку это требует серьезной модификации уже сформированных пакетов сетевого трафика.

Поскольку проводить тестирование целевой системы вручную очень трудоемко, в последнее время стали появляться автоматизированные средства эксплойтинга. Такие программы позволяют в автоматическом режиме обнаружить уязвимости целевого хоста, а затем провести ряд атак для эксплуатации этих уязвимостей. Общий алгоритм работы таких средств состоит из следующих шагов:

1. Сканирование портов и идентификация сервисов на исследуемых объектах.

2. На основе базы знаний выдвигается предположение о наличие уязвимостей в обнаруженных сервисах.
3. Проверка возможности эксплуатации предполагаемых уязвимостей.

Рассмотрим самые популярные средства автоматического аудита безопасности:

- Core Impact [14] – это программный продукт для проведения тестирования несанкционированных проникновений в систему. Проверяет исследуемые системы на наличие уязвимостей. С помощью него можно протестировать безопасность веб-приложений, сетевых систем, конечных устройств, мобильных устройств, беспроводных сетей и так далее. Позволяет моделировать многоступенчатые атаки и взаимодействовать с успешно атакованной системой. Имеет свою БД эксплойтов, которая ежемесячно пополняется собственной командой разработчиков. Из недостатков можно отметить довольно высокую цену;
- CANVAS/D2/Nessus Bundle [15]. Сборка CANVAS (компании Immunity) с D2 Exploit Pack (компании Square Security) интегрированные со сканером обнаружения уязвимостей Nessus позволяют достигнуть аналогичного подхода, заложенного в продукте CORE IMPACT. В отличие от своего конкурента, под платформу CANVAS, помимо собственных эксплойтов, возможен импорт сторонних эксплойтов. Это позволяет CANVAS'у охватить гораздо больше уязвимостей для проведения атак по сравнению с другими решениями данного сегмента;
- SAINT Vulnerability Scanner && SAINTexploit [16]. Аналог интегрированного CANVAS и Nessus с тем лишь отличием, что у SAINT Vulnerability Scanner и SAINTexploit один производитель. Функциональность же практически полностью идентична;

Все описанные выше программы являются коммерческими, из бесплатных стоит выделить Armitage.

2.1 Armitage

Мощное средство, основанное на Metasploit, программный комплекс под названием Armitage. Armitage – это GUI приложение (по сути, графическая оболочка), которое упрощает и несколько автоматизирует работу с Metasploit Framework.

С помощью Armitage можно представлять исследуемые хосты в визуальном режиме, автоматически подбирать эксплойты для них, управлять успешно установленными сеансами и так далее (рисунок 3).

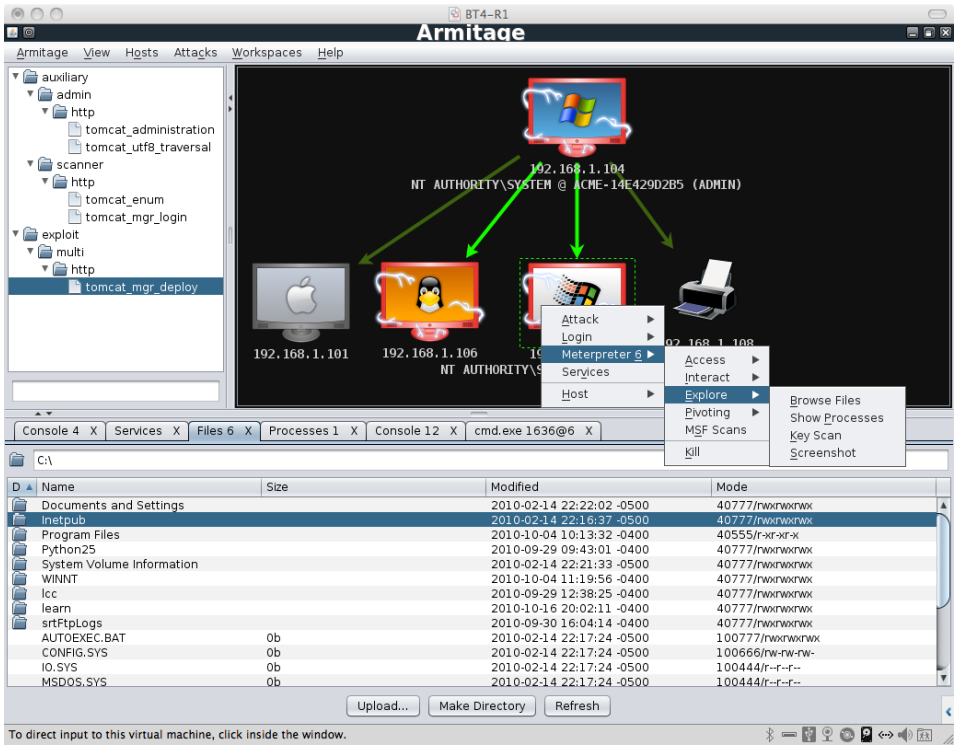

Рисунок 3 – Интерфейс Armitage

Для пользователей-экспертов доступна возможность удаленного управления и совместной работы с Metasploit (можно использовать один и тот же сеанс, иметь общую базу данных с информацией о хостах и их уязвимостях).

Обнаружение хостов в Armitage происходит с помощью средств сканирования, заложенных в Metasploit. Можно импортировать ранее составленный список хостов сети и запустить их сканирование, чтобы создать БД целей. Armitage

может представить эти данные в графическом режиме, таким образом, видно, какая из систем в данный момент анализируется, а к какой уже получен доступ и создано удаленное соединение.

Armitage автоматизирует выбор наиболее подходящих эксплойтов для данного хоста и выполняет их проверку (при наличии у эксплойта такой опции). Также можно запустить утилиту Hail Mary, которая запускает все имеющиеся эксплойты подряд.

При успешном получении доступа к удаленному хосту, Armitage помогает в выборе дальнейших действий. В состав Armitage входит ряд инструментов, основанных на специальном агенте Meterpreter. Meterpreter представляет собой агента, с помощью которого можно выполнять множество операций после проникновения в машину. Можно осуществить повышение прав в системе, загрузить список хэшей паролей в локальную БД, просмотреть файловую систему удаленной машины, запустить оболочку командной строки и прочее.

Кроме того, Armitage может создать так называемый «pivot» – «опорную точку», которая позволяет использовать захваченные хосты как платформы для организации дальнейших атак на другие машины и исследования сети.

3 Архитектура системы

На рисунке 4 показана схема автоматизированного тестирования:

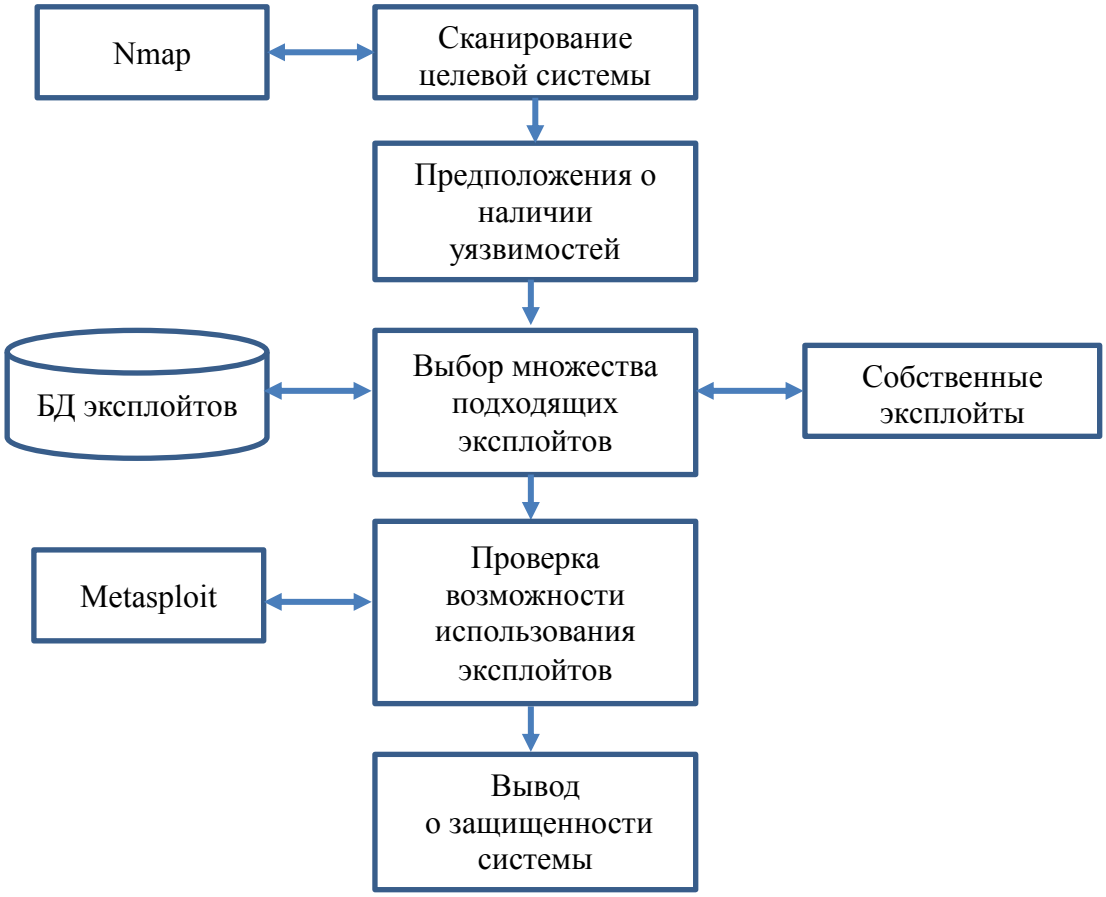

Рисунок 4 – Схема автоматизированного тестирования

Сначала происходит сбор сведений о целевой системе, затем выдвигаются предположения о наличии каких-либо уязвимостей в ней. После этого, из БД выбираются походящие эксплойты и проверяется их работоспособность. Если часть из них срабатывает, то система уязвима. Если же нет, то средства защиты успешно отражают атаки.

Разработанная система подразделяется на следующие взаимодействующие между собой компоненты (рисунок 5):

- Подсистема сбора информации (ПСИ)
- Подсистема формирования БД эксплойтов (ПФБДЭ)

- База данных анализатора (БДА)
- Metasploit Framework
- Подсистема анализа защищенности (ПАЗ)

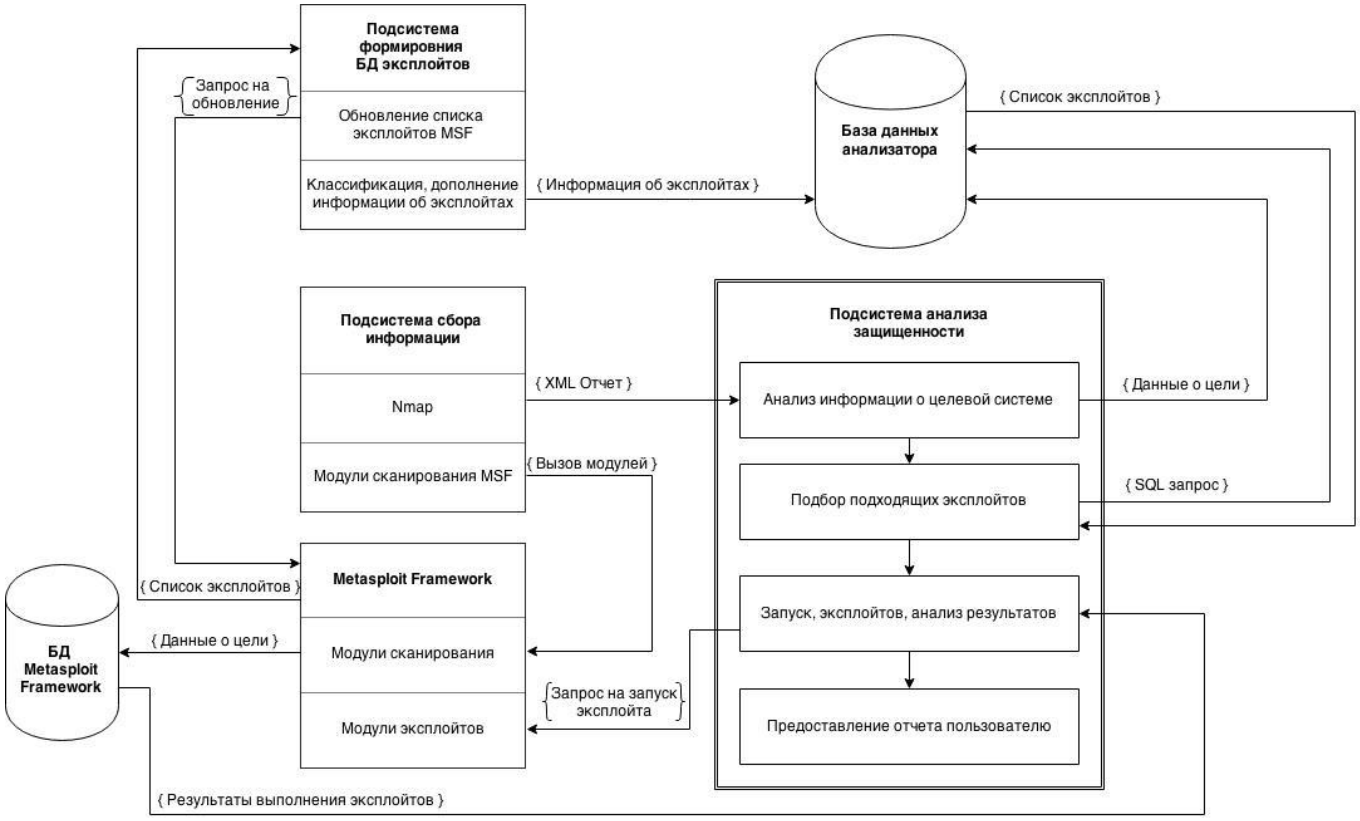

Рисунок 5 – Архитектура разработанной системы

3.1 Подсистема сбора информации

Задача данной подсистемы – сбор сведений о целевой системе. На вход подается IP-адрес «жертвы», после этого начинается сканирование. Основным инструментом этой подсистемы является сетевой сканер Nmap.

Nmap («Network Mapper») – это утилита с открытым исходным кодом, предназначенная для проведения подготовительных исследований сети или единичных целей и проверки их безопасности. Nmap посылает особые, «неправильные» ip пакеты для того, чтобы определить, доступен ли хост, какие порты открыты, какие службы (название, версия) запущены на них, какие средства защиты используются (брэндмауэры, СОВ, СПВ). Также, Nmap позволяет

определять операционную систему хоста (OS fingerprinting) вплоть до семейства, версии, языка, сервисного пакета и архитектуры.

На выходе Nmap выдает список хостов, которые были просканированы и дополнительную информацию, которую удалось извлечь, для каждого из них. Наиболее существенной информацией помимо версии ОС является список «значимых» портов. В нем содержится номер порта, имя службы, протокол и состояние порта. Состояние может принимать одно из следующих значений:

- open – открыт, служба на целевой машине прослушивает этот порт для установления соединения или принятия пакетов;
- filtered – фильтруется, значит, что брэндмауэр, фильтр пакетов или другое средство защиты блокирует доступ к этому порту и Nmap не может определить его состояние;
- closed – закрыт, никакое приложение не использует этот порт, однако он может быть открыт какой-нибудь службой в любой момент;
- unfiltered – не фильтруется, порты никак не реагируют на запросы от Nmap, следовательно, он не может определить, закрыты они или открыты.

Иногда Nmap выдает сочетание двух состояний, например, open | filtered или closed | filtered, когда невозможно детектировать, какое из этих состояний истинное.

Кроме этого, бывает, что Nmap’у удается распознать версию программного обеспечения, запущенного на хосте. Также при настройке определенных параметров можно определить DNS имя целевой машины, типы устройств, MAC-адрес и прочее.

Из различных доступных сканеров был выбран именно Nmap по ряду причин. Во-первых, удобное представление результатов в виде xml файлов [17], во-вторых, можно запускать напрямую из Metasploit Framework (db_nmap)[18]. Кроме того, по сравнению со своими аналогами, такими как p0f, XProbe 2, Hping, Amap, netcat и прочими, Nmap показывает самые корректные результаты и наиболее часто обновляется [19].

Рассмотрим основные опции сетевого сканера Nmap, которые были использованы при реализации программного комплекса [20].

- -sT (scan TCP) – используется метод сканирования TCP connect. Данный метод является одним из самым распространенных и заключается в попытке выполнения функции connect(), то есть в установлении соединения с портом на целевом хосте. Если функция выполнится успешно, это будет означать, что порт открыт и какая-то служба «слушает» его на стороне «жертвы». В противном случае, порт либо закрыт, либо блокируется сетевым экраном.
- Для использования метода не требуется иметь никаких привилегий, однако недостатком является то, что администратор сканируемого хоста может легко обнаружить факт сканирования, так как попытки соединения фиксируются в логах.
- -sS (scan SYN) – используется метод сканирования TCP SYN. Заключается в установлении неполного TCP соединения? то есть посылается TCP пакет с флагом SYN, и ожидается ответный пакет. Если в нем будут установлены флаги SYN и ACK, то порт открыт, если же RST флаг, то порт закрыт. При получении ответа с SYN + ACK, целевой системе посылается пакет с флагом RST для предотвращения открытия соединения.
- Необходимы привилегии администратора (root), однако, такой тип сканирования гораздо сложнее обнаружить.
- -sP (scan Ping) – посылаются ICMP пакеты с Echo запросами. Применяется, если необходимо узнать лишь факт активности хоста. Большинство современных брэндмауэров блокирует входящие ping запросы, поэтому, если есть точная информация о том, что хост активен, но он не отвечает, следует использовать опцию –Pn, которая говорит, что проверка состояния хоста не требуется.
- -T<0-5> – параметр, отвечающий за то, как часто Nmap шлет пакеты хосту, какие между ними задержки. Значения могут быть от 0 до 5 (0 – очень редко, 5 – очень часто).
- -A – опция, говорящая, что необходимо провести определение версии ОС, служб и открытых портов. Включает в себя множество различных методов

сканирования (-sC –sV –O –traceroute), является довольно легко обнаруживаемым режимом Nmap.

- -v – увеличение детальности логирования операций, выполняемых Nmap’ом.
- -sU (scan UDP) – производится UDP сканирование. Поскольку Udp – ненадежный протокол, с помощью этого метода не так легко определить статус порта. Посылается UDP пакет с пустым заголовком и, если порт закрыт, то должен вернуться ответ с ошибкой ICMP port unreachable. Если порт открыт, то не гарантируется, что ответ будет в принципе.
- -sV (scan Version) – сканирование с целью определить версию запущенных служб.
- -p – можно указать, какие порты необходимо просканировать. По умолчанию Nmap сканирует всего 1000 портов (0-1000).
- -O – определение ОС. Метод основан на нескольких различных факторах, например на значении поля TTL (Time To Live) полученного пакета. Для ОС Windows обычно это поле равно 128, в то время, как для ОС Unix, чаще всего выставляется значение 64. Также анализируются баннеры успешно определенных служб, проводится сигнатурный анализ и так далее.

Результаты простейшего сканирования утилитой Nmap выглядят следующим образом (рисунок 6):

Рисунок 6 – Пример результата выполнения Nmap

Помимо перечисленных выше методов сканирования портов, существует еще множество других [21]. В разработанном программном комплексе возможно использование всех техник сканирования, предлагаемых Nmap.

Стоит отметить, что в зависимости от ситуации, стоит применять различные типы сбора информации («разведки»). Обычно выделяют активную и пассивную «разведку».

При проведении пассивного сбора информации либо не происходит никакого взаимодействия с целевой системой, либо взаимодействие осуществляется строго по заданным заранее правилам. Все, чем располагает нарушитель – это какая-либо общедоступная информация о хосте и перехваченный трафик, отправленный «жертвой». Поэтому, пассивные методы направлены на различный анализ этого трафика. Из явных преимуществ следует указать абсолютную скрытность таких методов, то есть администратор исследуемого хоста не может обнаружить сам факт сканирования. Однако, подобные техники сильно ограничены и не могут предоставить всю интересующую информацию о системе.

Напротив, при активном сканировании, происходит отправка всевозможных запросов на атакуемый хост и анализ полученных ответов. В результате можно получить более подробную и достоверную информацию, однако велика вероятность обнаружения и блокирования межсетевым экраном.

Также Nmap имеет несколько различных способов обхода средств защиты информации, таких как сетевые экраны и антивирусы.

Зачастую, при попытке сканирования цели, возникают сложности, связанные с наличием на сканируемом хосте каких-нибудь программ, блокирующих трафик Nmap. Для того чтобы обойти эту защиту, можно воспользоваться одним из способов, предлагаемых ниже [22]:

- фрагментация пакетов (-f), может помочь, если на исследуемом хосте установлен устаревший брэндмауэр. Nmap дробит исходный пакет на части, таким образом, удается обойти средства защиты, основанные на сигнатурном поиске вредоносных пакетов;
- задание MTU – максимального размера полезного блока данных одного пакета (--mtu number target). Пакеты, с данными такого размера передаются без фрагментации. Если же данных больше, пакет разбивается на части;
- использование фальшивых адресов (-D decoy1,decoy2,dec). Если указать данный параметр, то реальный адрес отправителя (атакующего) будет заменен на один из указанных. Такая подмена сильно затруднит поиск нарушителя;
- сканирование Idle Zombie (-sI [Zombie IP] [Target IP]) – использование другого хоста в качестве своеобразного прокси-сервера. То есть такой тип сканирования позволяет посылать пакеты для исследования целевого хоста с другой машины. Таким образом, реальный IP-адрес нарушителя не будет присутствовать в логах брэндмауэра;
- указание номера порта отправителя (–source-port) – часто, при настройке сетевых экранов, администраторы разрешают весь входящий трафик, приходящий с определенного порта, например с 20, 53 или 67. Можно воспользоваться подобной оплошностью и сконфигурировать Nmap таким образом, чтобы он слал пакеты именно с таких портов;

- добавление случайных данных (–data-length) – многие брэндмауэры при проверке пакетов обращают внимание на их размер, пытаясь определить, не происходит ли попытки сканирования портов хоста, так как обычно большинство сканеров посылают пакеты одинакового размера. Данную опцию можно использовать, чтобы каждый раз посылать пакеты различной длины;
- подмена MAC-адреса (–spoof-mac) – данная техника эффектина, если на целевой системе настроена фильтрация по MAC-адресам. Можно выбирать производителя, а Nmap будет отправлять пакеты с соответствующими ему MAC-адресами;
- неправильная контрольная сумма (–badsum IP) – обычно контрольная сумма используется для проверки целостности пакетов. Однако, отправка пакетов с некорректной контрольной суммой, может помочь получить дополнительную информацию от системы.

Помимо получения данных о целевой системе с помощью средств Nmap, подсистема сбора информации использует некоторые возможности Metasploit Framework для подтверждения этой информации.

В MSF есть набор модулей, предназначенных для проведения сканирования удаленных хостов [23, 24]. Следующие модули являются самыми распространенными и применяются в реализованной подсистеме:

- SYN сканирование – для данного типа сканирования используется модуль auxiliary/scanner/portscan/syn;
- TCP сканирование – модуль auxiliary/scanner/portscan/tcp;
- SMB Version сканирование – определение версии ОС на основании информации о службе smb. Модуль - auxiliary/scanner/smb/smb_version;
- MS SQL сканирование – auxiliary/scanner/mssql/mssql_ping. Определение открытых портов, используемых Microsoft SQL Server.

Для каждого сканера можно задавать различные параметры, такие как IP-адрес исследуемого хоста, интерфейс, порты, которые нужно просканировать и так далее.

Отчет о сканировании Nmap выгружается в формате XML, затем данные из этого отчета группируются и заносятся в базу данных для дальнейшего использования. Результаты сканирования модулями MSF заносятся в БД автоматически.

3.2 Подсистема формирования БД эксплойтов

Основная задача данного компонента – составление базы данных эксплойтов на основе списка эксплойтов MSF. Работу этой подсистемы можно описать следующими шагами:

- Обновление списка эксплойтов MSF по Internet. Серверу Metasploit отправляется запрос с проверкой доступных обновлений эксплойтов. В случае если такие обновления присутствуют, происходит загрузка и распаковка новых эксплойтов.
- Определение портов и протоколов. Для каждого эксплойта, доступного для запуска через Metasploit осуществляется поиск соответствующих портов и протоколов. Поскольку любой эксплойт направлен на уязвимость в какой-то конкретной службе, можно определить, какой порт соответствует данному эксплойту. Для этого используется документ «Service Name and Transport Protocol Port Number Registry RFC 6335» [25], в котором приведен список сервисов и соответствующих им портов. Также этот список дополняется информацией из утилиты Nmap (Well Known Port List) [26]. Это файл «nmap-services», который содержит перечень названий портов, их номеров и протоколов. Также для каждого порта есть данные о том, с какой вероятностью он будет открыт на исследуемой системе. Данный список регулярно обновляется и содержит довольно актуальную информацию. В результате, для каждого эксплойта, выбранного из Metasploit происходит попытка поставить в соответствие порт и протокол. Если для эксплойта не прописано в явном виде, на какую службу он направлен, производится автоматический анализ его подробного описания, и определение службы. Затем подбор портов повторяется.

- Кроме того, часть портов определяется при помощи парсинга исходного кода эксплойтов. В некоторых из них содержится указание портов, используемых эксплойтами по умолчанию.
- Классификация эксплойтов. После получения данных об используемых эксплойтами портах, осуществляется группировка эксплойтов по типу операционной системы, для которой они предназначены (Windows, Linux, Unix) и по службам.
- Запись в БД. Вся доступная информация об эксплойтах (ОС, служба, имя, порты, протоколы, ранг, описание и прочее) заносится в БД PostgreSQL.

Таким образом, подсистема формирования БД эксплойтов занимается созданием, обновлением и поддержкой БД эксплойтов, которая содержит всю необходимую информацию для выборки подходящих эксплойтов для определенной цели.

3.3 База данных анализатора

Хранит основную информацию об эксплойтах, проверяемом хосте, открытых портах, службах и так далее. Главная цель данной БД – предоставлять доступ к данным обо всех доступных эксплойтах. Из нее можно получить список эксплойтов, пригодных, например, для атаки хоста с ОС Windows XP Professional SP2, с открытыми портами 80 и 445. Кроме того, можно узнать номер уязвимости, которую использует тот или иной эксплойт, ссылку на описание этой уязвимости в различных глобальных базах уязвимостей (таких как CVE), является ли он активным или пассивным, какой у него ранг и прочую полезную информацию.

3.4 Metasploit Framework

Metasploit Framework используется как источник эксплойтов и как интерфейс для их конфигурирования, запуска, а также анализа результатов их выполнения.

Взаимодействие с MSF осуществляется в автоматическом режиме, через консоль MSF.

3.5 Подсистема анализа защищенности

Подсистема анализа защищенности является самой важной в данном программном комплексе. В ее задачи входит взаимодействие с остальными подсистемами (подсистема сбора информации, формирования БД эксплойтов, MSF) и с БД, в которой хранятся данные о целевой системе и эксплойты, анализ информации об исследуемом хосте, полученной от подсистемы сбора информации, составление модели этой системы, выбор подходящих эксплойтов, построение сценариев атак, отправка команд на запуск эксплойтов в MSF, оценка результатов выполнения атак, принятие решения о последующих действиях и так далее.

Рассмотрим более детально работу данной подсистемы. Все шаги выполняются в автоматическом режиме, то есть участие пользователя не требуется.

Первым этапом работы подсистемы анализа защищенности является сканирование целевого хоста для формирования его модели. Для этого, ПАЗ посылает команду ПСИ с указанием IP-адреса цели. ПСИ с помощью утилиты Nmap и модулей сканирования MSF проводит сканирование цели и возвращает отчет с результатами, содержащий описание версии ОС хоста, установленного сервисного пакета, языка, и других данных, а также список открытых портов и соответствующих им служб. Всю эту информацию ПАЗ заносит в свою БД, в таблицы с определенной структурой.

Параллельно с этим шагом возможен запуск обновления БД с эксплойтами с помощью подсистемы формирования БД эксплойтов. Происходит соединение с сервером Metasploit и загрузка обновленного списка эксплойтов. Затем, ПФБДЭ выполняет сопоставление эксплойтов и портов и создает БД эксплойтов со всей доступной информацией о них.

После того, как модель атакуемого хоста составлена, и БД эксплойтов содержит актуальную информацию, можно переходить к подбору эксплойтов. Для

этого ПАЗ генерирует сложный SQL-запрос к БД и в ответ получает выборку с подходящими для конкретной системы эксплойтами. При выборе эксплойтов учитываются следующие параметры:

- ОС целевого хоста, ее версия, язык, сервисный пакет и любая другая информация о нем, полученная на этапе сканирования.
- номера портов на хосте, имеющих статус «open». Подобный статус означает, что какая-то служба запущена и готова к приему пакетов по данному порту. А значит, если в этой программе есть известная уязвимость и в БД содержится эксплойт, написанный для нее, то можно успешно провести атаку на систему.
- Имя и версия службы, запущенной на открытом порте. По этим данным также можно подобрать эксплойты.

В результате этих действий ПАЗ определяет ограниченный список эксплойтов, теоретически соответствующих анализируемой системе. Далее необходимо упорядочить их и проверить, какие действительно сработают и получат доступ к системе, а какие нет.

Сортировка эксплойтов происходит по их рангу, присвоенному Metasploit. Для анализа берутся только те эксплойты, ранг которых выше или равен 400 (GoodRanking), то есть это те эксплойты, для которых есть сведения о системах и ПО, на которые они нацелены. Эксплойты с более низким рангом брать нецелесообразно, так как они успешно срабатывают лишь в редких случаях.

Помимо ранга учитывается «агрессивность» эксплойта. Все эксплойты в MSF делятся на активные и пассивные [27]. В случае с пассивными эксплойтами, необходимо, чтобы выполнилось одно или несколько условий на стороне атакуемого хоста, чтобы уязвимость сработала. То есть эксплойт будет ожидать, пока пользователь на машине «жертве» выполнит определенные действия. Такие эксплойты ПАЗ не использует по причине того, что время их выполнения очень непредсказуемо, а анализ системы желательно провести прямо сейчас, а не дожидаться, пока пользователь, например, откроет браузер. Активные же эксплойты срабатывают (или нет) сразу после их запуска. Именно они еще и называются

«агрессивными». Подобное ограничение на характер эксплойтов вызван тем, что реализуемая система должна быть полностью автоматизированной.

Далее, когда сформирован отсортированный список эксплойтов для анализируемой системы, начинается процесс их конфигурации. Для каждого эксплойта задаются необходимые параметры, опции, настройки. Основным параметром любого эксплойта является его полезная нагрузка.

Полезная нагрузка – это часть эксплойта, которая выполняет действия, ради которых он и запускался, например, копирование данных, удаление данных, загрузка ВПО, открытие соединения с зараженной машиной, запуск оболочки ОС (командного интерпретатора) и другие. Было принято решение в качестве полезной нагрузки для всех эксплойтов выбирать Meterpreter [28].

Meterpreter (MP) – это расширяемая, гибкая, унифицируемая полезная нагрузка, которая предоставляет широкие возможности пост-эксплуатации системы в случае удачного запуска эксплойта. Обычно в качестве полезной нагрузки выступают шеллкоды, однако Meterpreter имеет ряд преимуществ. Во-первых, при запуске оболочки («шелла») ОС создается отдельный процесс, следовательно, его легко обнаружить. Во-вторых, большинство СОВ довольно эффективно находят шеллкоды по трафику, передаваемому ими атакующему, так как все команды пересылаются в открытом виде. В-третьих, процесс может быть ограничен командой chroot. Chroot – это операция изменения корневого каталога в Unix-подобных ОС. Программа, запущенная с измененным корневым каталогом, будет иметь доступ только к файлам, содержащимся в данном каталоге [29]. Наконец, оболочки сильно разнятся в зависимости от ОС, как по формату команд, так и по набору возможностей. Разработчики Meterpreter решили все эти проблемы.

Проблема с порождением нового процесса отсутствует, так как Meterpreter является многоступенчатым шеллкодом. После успешного выполнения эксплойта осуществляется загрузка MP в виде DLL библиотеки. Затем он размещается в адресном пространстве процесса, уязвимость которого была проэксплуатирована. Далее происходит запуск MP на исполнение в виде нового потока. Таким образом,

MP и его расширения не создают новый процесс, а работают в контексте проэксплуатированного процесса в виде DLL библиотеки.

Поскольку MP включает в себя достаточно много возможностей по взаимодействию с ОС, например, загрузку и выгрузку файлов, изменение файловой системы и реестра и другие, трудностей с chroot и доступностью стандартных функций и программ не возникает. Также есть возможность загружать собственные DLL библиотеки с реализованными функциями, которые потребовались в каком-то конкретном случае.

Еще одна отличительная особенность Meterpreter – возможность миграции с процесса на процесс [30]. Таким образом, например, при закрытии пользователем атакуемой машины процесса, в контексте которого работал MP, достаточно послать команду «migrate» и произойдет смена контекста. Можно также мигрировать в какой-нибудь системный процесс, например, explorer.exe, тогда вероятность того, что он завершится, сильно уменьшится. Поскольку общение с MP происходит все время через один и тот же сокет, соединение не будет прервано. В случае с шеллкодами, возможности миграции нет.

Поскольку MP работает только с памятью процесса и не осуществляет записи на жесткий диск компьютера, многим СОВ и антивирусам обнаружить его не удается. Кроме того, команды MP передает в зашифрованном виде, поэтому детектировать их не так просто.

Кратко перечислим основные команды Meterpreter [31]:

- cat – вывод содержимого файла;
- clearev – очистка журналов Windows (Application, System и Security);
- download – скачивание файла с удаленной системы;
- edit – изменить файл;
- execute – выполнить команду;
- getuid – вывести пользователя, от имени которого запущен MP;
- hashdump – скопировать содержимое БД Диспетчера учетных записей (SAM);

- ipconfig – показать сетевые интерфейсы и адреса;
- ps – список запущенных процессов;
- shell – запуск стандартной оболочки ОС;
- upload – загрузить файл на удаленную систему;
- webcam_list – список доступных видеокамер на целевом хосте;
- webcam_snap – сделать снимок с видеокамеры;

После задания полезной нагрузки всем эксплойтам, осуществляется последовательная проверка их работоспособности.

Для этого сначала выполняется команда check. Не все эксплойты поддерживают данную опцию, однако, для остальных можно проверить, содержит ли целевая система уязвимость, на которую направлен эксплойт, не запуская его. Таким образом, можно отбросить часть эксплойтов, которые точно не выполнятся.

Далее подается команда MSF и он пытается выполнить оставшиеся эксплойты на целевой системе. При окончании работы эксплойта (успешном или нет) создается новая запись в таблице в БД MSF. По значению в определенном поле этой таблицы можно определить, сработал ли эксплойт или нет. Результат запоминается анализатором и запускается следующий.

Таким образом, по завершении работы ПАЗ можно узнать, сколько и какие эксплойты успешно выполнились и получили доступ к системе, а сколько и какие - нет. Статистика выводится пользователю, с указанием уязвимостей, которые необходимо закрыть на целевой системе.

Кроме того, пользователь может отдельно запустить любой из сработавших эксплойтов и воспользоваться возможностями полезной нагрузки – Meterpreter. Автоматизированная работа ПАЗ с MP не предусмотрена, так как цель данного комплекса – тестирование средств защиты информации, то есть достаточно знать сам факт успешного выполнения эксплойта, пост-эксплуатация не требуется.

4 Разработка программного комплекса

Для реализации программного комплекса были выбраны следующие инструменты:

- ОС Kali Linux – ОС, специально предназначенная для проведения тестов на проникновение, включает в себя огромное множество предустановленных программ и утилит для программно-технической экспертизы, таких как Aircrack-ng, Burp_suite, Hydra, John_the_Ripper, Metasploit, Nmap, Wireshark, Armitage и другие. Поставляется в виде ISO-образа и образа для VMWare [32];
- VMWare player – бесплатный программный продукт, предназначенный для запуска образов виртуальных машин. Использовался для установки Kali Linux и других ОС, выступавших в роли целевых систем;
- Metasploit Framework – бесплатный продукт Metasploit, с помощью которого можно осуществлять запуск эксплойтов;
- Nmap – утилита, предназначенная для сканирования.IP-сетей;
- PostgreSQL – свободно распространяемая объектно-реляционная система управления базами данных. Поскольку MSF использует именно эту СУБД, для упрощения экспорта/импорта таблиц, было решено для подсистемы анализа защищенности использовать так же PostgreSQL;
- Java – в качестве языка программирования была выбрана Java.

4.1 Metasploit

При реализации программного комплекса для анализа защищенности АС использовался Metasploit Framework. С помощью него происходит запуск эксплойтов и дальнейшая эксплуатация уязвимостей.

Metasploit Project – это проект, созданный для предоставления информации об уязвимостях, предоставляющий средства для создания сигнатур для систем обнаружения вторжения (СОВ), написания и тестирования эксплойтов.

Самый известный продукт – Metasploit Framework (MSF) – бесплатная платформа, предназначенная для создания, отладки и запуска эксплойтов. Помимо этого, проект включает в себя базу опкодов и шеллкодов.

В настоящее время MSF переписан на язык Ruby (предыдущие версии были написаны на Perl’e) и принадлежит компании Rapid7, которая специализируется на средствах защиты информации.

Возможности MSF довольно широки. Платформа предоставляет инструмент для создания, тестирования и выполнения эксплойтов. Для выбранного конкретного эксплойта можно задать полезную нагрузку (payload), в зависимости от которой, в случае успешного выполнения эксплойта, будет совершено то или иное действие в атакуемой системе, например, установка shell или VNC сервера (система удалённого доступа к рабочему столу компьютера). Также можно зашифровать шеллкод, чтобы скрыть атаку от СОВ и СПВ. Metasploit Framework совместим с некоторыми утилитами-сканерами, такими как Nmap и Nessus. Можно загружать в MSF отчеты с результатами сканирования хостов и использовать эту информацию для выбора подходящих эксплойтов.

Самый базовый сценарий атаки с помощью MSF состоит из следующих шагов (рисунок 7):

1. Выбор и конфигурирование эксплойта.
2. Проверка пригодности данного эксплойта для целевой системы (опционально, доступно не для всех эксплойтов).
3. Выбор и настройка полезной нагрузки.
4. Выбор алгоритма шифрования, чтобы СОВ не обнаружила атаку.
5. Выполнение эксплойта.

Metasploit Framework имеет модульную структуру: эксплойты, полезные нагрузки, сетевые сканеры, – все это модули. Это позволяет сочетать любые модули друг с другом (например, для эксплойта можно выбрать одну из сотен вариантов полезной нагрузки).

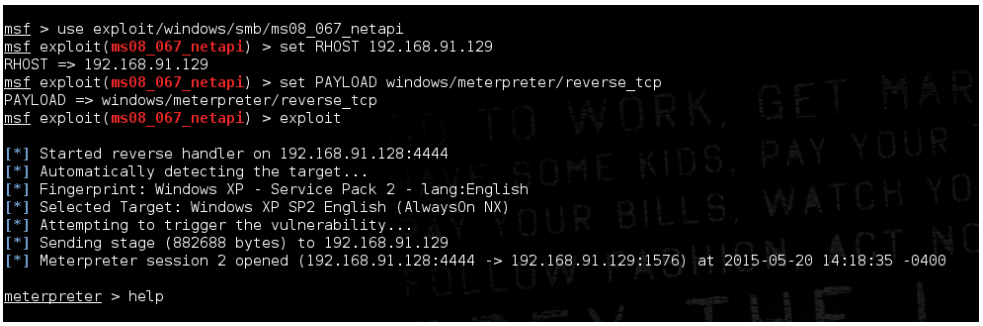

Рисунок 7 – Пример успешного запуска эксплойта

Metasploit можно установить на Unix (в том числе и на Linux и Mac OS X) и на Windows.

Существует несколько интерфейсов Metasploit. Самый популярный и бесплатный – Metasploit Framework, который состоит из командной строки, позволяющей вручную настраивать и запускать эксплойты. Он и был выбран в качестве средства взаимодействия с целевой системой.

Приведем список основных команд в MSF:

- search <keyword>: при запуске команды search без указания ключевых слов, выводится список всех доступных эксплойтов. В качестве параметров можно передавать различные свойства эксплойта, такие как имя, путь, тип, автора и так далее;
- show exploits: список всех доступных на данный момент эксплойтов;
- show payloads: аналогично предыдущим командам show, показывает доступные полезные нагрузки;
- show options: опции, которые можно использовать для данного эксплойта или полезной нагрузки;
- info <type> <name>: вывод подробной справочной информации об эксплойте;
- use <exploit_name>: указание MSF, какой эксплойт использовать;
- set <OPTION> <value>: данная команда используется для задания значения параметра (опции) данного эксплойта, например IP-адрес целевой системы

или имя полезной нагрузки, которую планируется использовать при успешном выполнении эксплойта;

- exploit: запуск выбранного эксплойта;
- help: команда help выдает полный перечень всех доступных команд системы.

Metasploit Framework отличается от своих аналогов следующими преимуществами:

- написан на Ruby, а значит, является кроссплатформеннным;
- предпакетная поддержка расширяемых утилит, библиотек и других возможностей, таких как отладка, кодирование, логирование, временные задержки, дополнение до нужного размера NOP'ами и SSL [33];
- понятная, интуитивная, модульная и расширяемая среда и API эксплойтов;
- высоко оптимизированная мультиплатформа, автоматически загружаемые полезные нагрузки;
- усовершенствованный обработчик и поддержка «callback», которая сокращает код эксплойта;
- поддержка различных сетевых опций и протоколов, которые могут использоваться для разработки зависимого от протокола кода;
- открытый исходный код, поддерживаемый сообществом разработчиков;
- поддержка расширенных возможностей и сторонних утилит, таких как UploadExec, InlineEgg, Impurity.

Помимо Metasploit Framework есть еще несколько интерфейсов Metasploit, таких как Metasploit Community, Metasploit Express, Metasploit Pro. Они представляют собой веб-интерфейсы для MSF с различной функциональностью.

4.1.1 Ранжирование эксплойтов в Metasploit.

Начиная с MSF версии 3.3..2, каждому модулю с эксплойтом присвоен ранг, основанный на потенциально возможном влиянии на целевую систему [34].

Значения рангов могут быть следующими:

- ExcellentRanking – эксплойты никогда не нарушают работоспособность сервиса. К таким эксплойтам относятся SQL инъекции, RFI, LFI. Никаким эксплойтам, вызывающим повреждение память, не должен быть присвоен такой ранг. Возможны редкие исключения (WMF Escape());
- GreatRanking – эксплойты предназначены для конкретных систем, и либо умеют автоматически проверять, подходит цель для данного эксплойта или нет, либо используют специфичный для приложения адрес возврата после проверки версии этого приложения;
- GoodRanking – эксплойты предназначены для конкретных систем и конкретного ПО (например, English, Windows XP for a desktop app, 2003 for server);
- NormalRanking – эксплойты достаточно надежны, однако, зависят от конкретной версии ПО и не могут автоматически определять эту версию;
- AverageRanking – эксплойты ненадежны и сложно эксплуатируемы;
- LowRanking – эксплойты практически невозможно использовать (менее 50% успешных запусков) для распространенных платформ;
- ManualRanking – эксплойты нестабильны или сложны в использовании, обычно представляют собой DOS атаки. Также этот ранг используется, когда эксплойт не может быть использован без настройки пользователем (например, php_eval).

4.1.2 Типы модулей в MSF

В Metasploit все (скрипты, файлы, программы) является модулями. Выделяют 6 типов модулей:

- auxiliary – модули, помогающие атакующему выполнять различные задачи, такие как сканирование портов, определение версий, анализ сетевого трафика;

- exploit – модули, содержащие эксплойты, то есть код, использующий какую-нибудь уязвимость в системе и позволяющий выполнить полезную нагрузку, например, переполнение буфера или обход аутентификации;
- payload – модули, содержащие полезную нагрузку, то есть то, что должно быть исполнено сразу после успешного выполнения эксплойта, например, установление удаленного соединения, запуск сеанса meterpreter или исполнение каких-либо системных команд;
- post – различные программы, которые могут быть запущены после успешного эксплойтирования и установления удаленного соединения, например, сбор паролей, установка программ-шпионов, скачивание файлов и так далее;
- encoder – программы, выполняющие шифрование полезной нагрузки для защиты от обнаружения защитными средствами;
- nop – генераторы NOP'ов. NOP – это инструкция на языке Ассемблер, которая ничего не делает. Машинный код данной инструкции различается для каждого типа архитектуры системы. Обычно NOP инструкции используются для приведения размера исполняемых файлов к определенному размеру.

MSF можно использовать во при проведении пентестов для создания отчетов вместе с прочими системами автоматического обнаружения уязвимостей. При помощи Metasploit можно определить, являются ли найденные уязвимости действительно опасными и можно ли их использовать для получения доступа к системе.

Кроме того, MSF можно использовать для тестирования новых эксплойтов. Для этого нужно настроить локальный сервер с уязвимостью, которую использует эксплойт. Таким образом, можно быстро проверить эффективность созданного эксплойта.

Также, MSF прекрасно подходит для проверки корректности настроек различных СОВ на случай сетевых атак.

Чтобы начать работать с Metasploit Framework, необходимо запустить две службы: postgresql и metasploit. Службы запускаются командой «service

<имя_службы> start». Для того чтобы можно было легко стартовать их из основной программы, был написан небольшой скрипт:

```
#!/bin/bash
service postgresql start
service metasploit start
```

Создание каких-либо сторонних процессов в Java осуществлялось с помощью класса ProcessBuilder [35]. Так, например, чтобы запустить приведенный выше скрипт, выполняются следующие команды:

```
ProcessBuilder pb = new ProcessBuilder("/root/Diploma/services.sh");
Process process = pb.start();
process.waitFor();
```

Запуск этого скрипта происходит каждый раз при старте программного комплекса.

Процесс реализации был также разделен по подсистемам программного комплекса. Начнем описание с подсистемы формирования БД эксплойтов.

4.2 Подсистема формирования БД эксплойтов

Первым шагом при создании БД эксплойтов является обновление списка эксплойтов Metasploit Framework. Процесс обновления заключается в обновлении самого MSF и выполняется при помощи команды «msfupdate» [36]. Вызов команды опять же происходит через методы класса ProcessBuilder.

После того, как MSF обновился, из него выгружается список эксплойтов. Делается это следующим образом: поскольку эксплойты хранятся в определенной директории («/usr/share/metasploit-framework/modules/exploits/<название_ОС>/<название_службы>/»), можно рекурсивно открывать папки с эксплойтами и заносить в таблицу их имена, ОС и службы, для которых они предназначены.

Кроме того, для каждого эксплойта происходит попытка сопоставить ему порт и протокол. Для этого подготавливается специальный файл со списком служб и соответствующих им портов и протоколов.

Файл формируется из документа «Service Name and Transport Protocol Port Number Registry». В нем содержатся следующие данные: имя службы, номер порта, протокол транспортного уровня, описание службы. Помимо этого документа, файл дополняется записями из файла «nmap-services», используемого утилитой Nmap. Он также содержит перечень портов, служб и протоколов. Его структура довольно проста [26]. В нем есть три колонки, разделенные пробелами. В первой хранится имя службы, во второй – номер и протокол, разделенные знаком «/», в третьей – вероятность того, что данный порт будет открытым на исследуемой системе.

В результате, из двух источников собирается один, общий файл «services.txt». Всего в нем около 12000 записей. Ниже приведен фрагмент этого файла (рисунок 8):

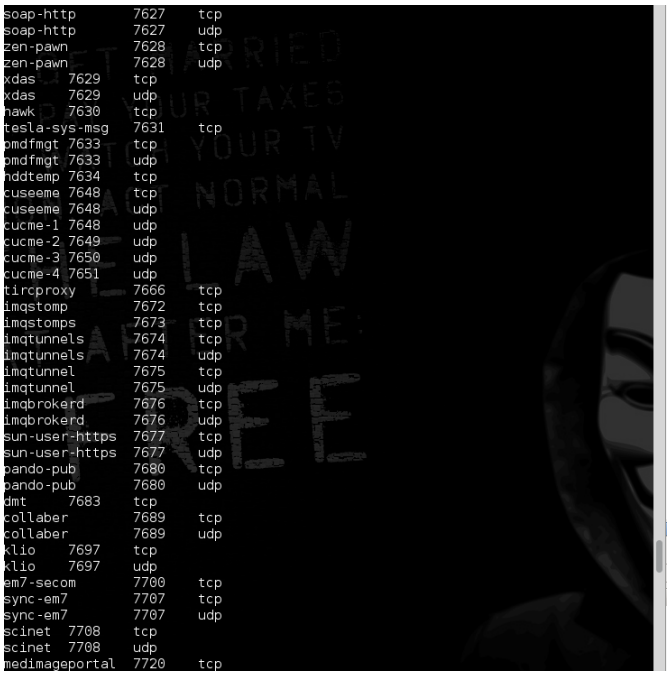

Рисунок 8 – Фрагмент файла «services.txt»

По этому файлу и происходит определение портов для эксплойтов.

Таким образом, после выполнения всех этих действий, получается таблица со столбцами: «ОС», «Служба», «Имя_эксплойта», «Порт», «Протокол». Затем информация о каждом эксплойте дополняется из таблиц MSF – «module_details» и «module_platforms». В них хранятся ранги эксплойты, их описания, типы (активный/пассивный), платформы, для которых они написаны и прочее. Далее,

полученная таблица заносится в БД эксплойтов (рисунок 9). Общение с СУБД PostgreSQL происходит при помощи драйвера JDBC [37]. Связь происходить по следующей схеме: имеется единый интерфейс, к которому подключается драйвер для работы с PostgreSQL, после этого можно передавать запросы БД.

	id integer	os text	service character vary	name text	fullname text	port character	protocol character	file text	refname text	textname text	rank integer	description text	privileged boolean	disclosure_dat timestamp with	default_ta integer	default text	stance character	ready boolean
391	1014	linux	http	mutiny_subnetmask_exec	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	Mutiny F	600	This mod	TRUE	2012-10-22	1		passive	TRUE
392	1014	unix	http	mutiny_subnetmask_exec	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	Mutiny F	600	This mod	TRUE	2012-10-22	1		passive	TRUE
393	974	php	http	nas4free_php_exec	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	NAS4Free	500	NAS4Free	TRUE	2013-10-30	0		aggressi	TRUE
394	990	windows	http	netwin_surgeftp_exec	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	Netwin S	400	This mod	FALSE	2012-12-06			aggressi	TRUE
395	990	unix	http	netwin_surgeftp_exec	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	Netwin S	400	This mod	FALSE	2012-12-06			aggressi	TRUE
396	1023	unix	http	op5_license	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	OP5 lice	600	This mod	TRUE	2012-01-05	0		aggressi	TRUE
397	1065	linux	http	op5_welcome	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	OP5 welc	600	This mod	TRUE	2012-01-05	0		aggressi	TRUE
398	1065	unix	http	op5_welcome	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	OP5 welc	600	This mod	TRUE	2012-01-05	0		aggressi	TRUE
399	1044	java	http	openfire_auth_bypass	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	Openfire	600	This mod	TRUE	2008-11-10	0		aggressi	TRUE
400	1044	linux	http	openfire_auth_bypass	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	Openfire	600	This mod	TRUE	2008-11-10	0		aggressi	TRUE
401	1044	windows	http	openfire_auth_bypass	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	Openfire	600	This mod	TRUE	2008-11-10	0		aggressi	TRUE
402	983	unix	http	openmediavault_cmd_exec	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	OpenMedi	600	OpenMedia	TRUE	2013-10-30	0		aggressi	TRUE
403	983	linux	http	openmediavault_cmd_exec	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	OpenMedi	600	OpenMedia	TRUE	2013-10-30	0		aggressi	TRUE
404	1063	php	http	openx_backdoor_php	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	OpenX Ba	600	OpenX Ad	FALSE	2013-08-07	0		aggressi	TRUE
405	1056	java	http	opmanager_socialit_file_upload	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	ManageEr	600	This mod	TRUE	2014-09-27	0		aggressi	TRUE
406	1072	windows	http	oracle_reports_rce	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	Oracle F	500	This mod	FALSE	2014-01-15	0		aggressi	TRUE
407	1072	linux	http	oracle_reports_rce	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	Oracle F	500	This mod	FALSE	2014-01-15	0		aggressi	TRUE
408	1026	php	http	pandora_upload_exec	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	Pandora	600	This mod	FALSE	2010-11-30	0		aggressi	TRUE
409	981	php	http	php_cgi_arg_injection	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	PHP CGI	600	When run	FALSE	2012-05-03	0		aggressi	TRUE
410	970	php	http	php_volunteer_upload_exec	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	PHP Volu	600	This mod	FALSE	2012-05-28	0		aggressi	TRUE
411	1027	php	http	phpldapadmin_query_engine	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	phpLDAPa	600	This mod	FALSE	2011-10-24	0		aggressi	TRUE
412	1049	php	http	phpmoadmin_exec	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	PHPMoAdm	600	This mod	FALSE	2015-03-03	0		aggressi	TRUE
413	1040	php	http	phpmyadmin_3522_backdoor	exploit/multi/ht	80/8008	sctp/tcp	/usr/she	multi/ht	phpMyAdm	300	This mod	FALSE	2012-09-25	0		aggressi	TRUE

Рисунок 9 – Часть таблицы с эксплойтами.

При окончании формирования/обновления БД эксплойтов пользователю выводится статистика по эксплойтам (сколько их содержится в БД) (рисунок 10).

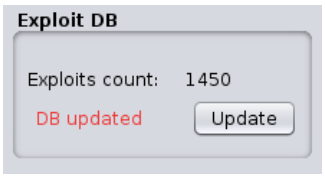

Рисунок 10 – Результат обновления БД эксплойтов

4.3 Подсистема сбора информации

На вход подсистеме подается IP-адрес целевой системы и профиль сканирования Nmap. Пользователь может выбрать из следующих вариантов (рисунок 11):

- Intense scan – довольно быстрое сканирование, проверяется большинство TCP портов, определяются запущенные службы;
- Intense scan plus UDP – добавляется UDP и TCP SYN сканирования;

- Intense scan, all TCP ports – проверка всех портов (с 1 по 65535). По умолчанию проверяется лишь первые 1000 портов;
- Intense scan, no ping – используется опция –Pn, то есть предполагается, что хост точно активен и нет необходимости посылать ему echo запросы;
- Quick scan – очень быстрое сканирование за счет того, что проверяется только 100 основных TCP портов;
- Quick scan plus – то же, что и «Quick scan», но еще добавляется распознавание версий запущенных служб;
- Regular scan – все параметры выставлены по умолчанию (TCP SYN сканирование, первые 1000 портов, ICMP echo запросы для определения состояния хоста);
- Slow comprehensive scan – самый подробный и длительный вариант сканирования.

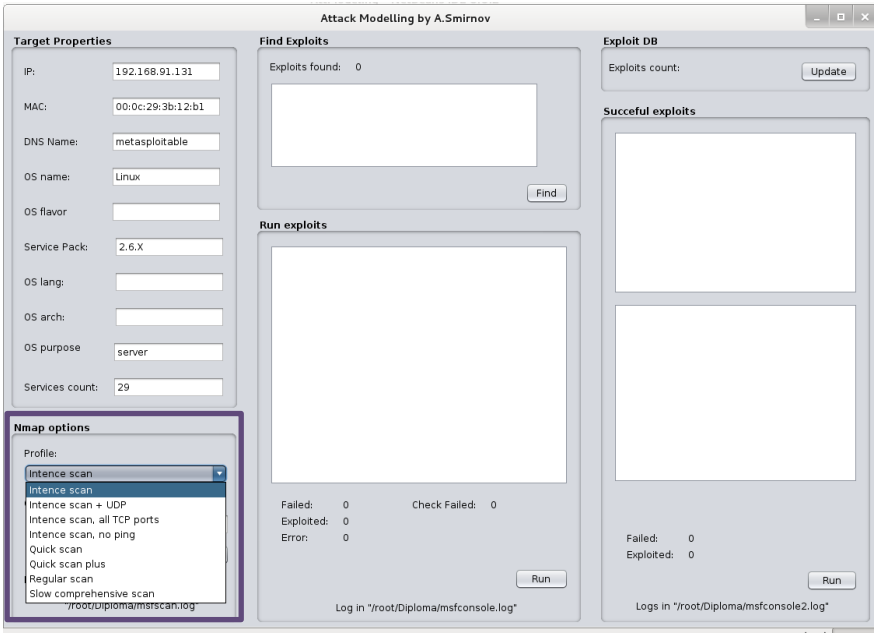

Рисунок 11 – Выбор режима сканирования Nmap

После выбора профиля для сканирования утилитой Nmap, формируется набор параметров Nmap. Например, для Intense scan он будет таким: «-T4 -A -v –O». Затем эти параметры подаются на вход Nmap и происходит запуск сканирования.

Данные с результатами (информацией о хосте, открытых портах и запущенных службах) заносятся в БД (рисунки 12, 13).

	id [PK] serial	created_at timestamp	address inet	mac character varying(255)	c	name character	state character	os_name character	os_flavor character	os_sp character	os_lang	arch character	workspace integer	updated_a timestamp	purpose text	ir	c	s	v	note_coun integer	vuln_coun integer	service_count integer	host_detai integer	exploit_att integer	cred_coun integer
1	15	2015-05-	192.168.	00:0c:29:9e:a2:a9		APO3WXPE	alive	Windows		SP2		x86	1	2015-05-	client					15	1	175	0	134	0

Рисунок 12 – Таблица с данными о хосте

42	46	15	2015-05-	135	tcp	open	msrpc	2015-05- Microsoft Windows RPC
43	47	15	2015-05-	139	tcp	open	netbios-	2015-05- ''
44	48	15	2015-05-	445	tcp	open	microsof	2015-05- Microsoft Windows XP microsoft-ds

Рисунок 13 – Таблица с портами и службами

После проведения сканирования Nmap, проводится исследование хоста средствами Metasploit. Производится запуск модуля auxiliary/scanner/portscan/tcp – TCP сканирование портов. Результаты так же заносятся в БД.

По завершении работы подсистемы сбора информации в окне программы заполняются поля, связанные с целевой системой (рисунок 14). Вся подготовительная работа на этом этапе сделана, можно переходить к основной части – поиск эксплойтов и попытка их использования.

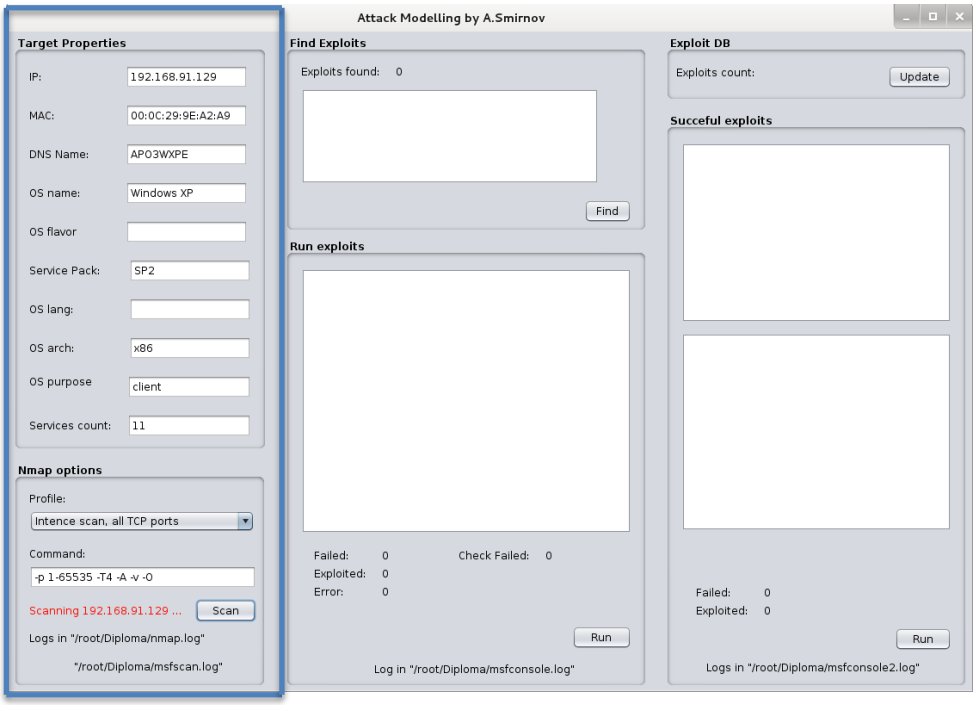

Рисунок 14 – Результаты работы ПСИ

4.4 Подсистема анализа защищенности

Подсистема анализа защищенности должна обмениваться командами с Metasploit Framework, а точнее – с командной строкой MSF – msfconsole. Для того, чтобы передать инструкцию консоли MSF производятся следующие действия:

- если набор команд и параметров заранее известен, то используется опция консоли «-r». Она позволяет подавать на вход msfconsole файл с командами, которые выполнятся сразу после запуска командной строки. Так, например, осуществляется вызов модуля TCP сканирования (формируется файл с командами, затем он передается в качестве параметра msfconsole);
- если же команды должны подаваться динамически (например, неизвестно, какая команда будет следующей), то сначала создается процесс msfconsole при помощи класса ProcessBuilder. Далее, ожидается завершение загрузки консоли MSF. Так как заранее невозможно предугадать, сколько времени это займет, программа через определенные промежутки времени подсчитывает размер файла, в который перенаправлен стандартный поток вывода msfconsole.

Как только размер перестает увеличиваться, msfconsole успешно загрузилась. После этого, посредством метода java.lang.Process.getOutputStream(), происходит получение потока вывода процесса. Затем, с помощью класса BufferedWriter создается буферизованный поток. Класс BufferedWriter записывает текст в поток, предварительно буферизируя записываемые символы, тем самым снижая количество обращений к физическому носителю для записи данных [38]. Теперь можно писать в этот поток команды и они будут передаваться консоли Metasploit.

Стоит отметить, что при создании любого процесса (Nmap, msfconsole) производится перенаправление его выходного потока в файл посредством метода redirectOutput класса ProcessBuilder. Таким образом, можно легко вести логирование всех запускаемых процессов.

Продолжим рассмотрение реализации ПАЗ. Первая составляющая этой подсистемы служит для подбора эксплойтов, подходящих под целевую систему. Первым шагом происходит установка соединения с БД анализатора при помощи

JDBC драйвера. Затем выполняется запрос к БД для получения списка открытых портов на целевом хосте. После того, как порты получены, можно сформировать SQL запрос для выборки подходящих эксплойтов. Отбираются эксплойты по следующим параметрам: во-первых, ОС цели должна совпадать с ОС, для которой написан эксплойт, во-вторых, в результирующую выборку попадают те эксплойты, у которых порт, на который они нацелены, один из открытых портов целевой системы. Кроме того, выбираются только активные эксплойты, то есть такие, которые для успешного срабатывания не требуют никаких действий со стороны пользователя исследуемой системы. Также, предпочтение отдается эксплойтам с более высоким рангом. Все отобранные эксплойты выводятся пользователю на экран.

После того, как составлен список потенциально опасных для системы эксплойтов, можно приступать к тестированию. За это отвечает вторая составляющая ПАЗ – функция запуска эксплойтов. Сначала происходит создание отдельного процесса консоли Metasploit. На вход msfconsole подается заранее созданный файл с инструкциями установки параметров эксплойтов. Чтобы не задавать параметры, например, RHOST – IP-адрес целевой системы или LHOST – IP-адрес атакующей системы, для каждого эксплойта отдельно, применяется специальная команда «setg» - задание глобальных параметров. Таким образом, достаточно задать все параметры эксплойтов один раз, при запуске консоли MSF. Кроме того, в зависимости от ОС, выбирается полезная нагрузка. Далее, последовательно в цикле для каждого отобранного эксплойта выполняется команда «use <имя_эксплойта>». Эта команда осуществляет переход к окружению эксплойта. После этого можно запускать эксплойт (команда «exploit»). Чтобы определить, когда эксплойт завершил свою работу (успешно или нет), производится подсчет записей в таблице exploit_attempts в БД Metasploit Framework. В эту таблицу Metasploit автоматически заносит все попытки выполнения эксплойтов. Если число записей увеличилось, значит эксплойт выполнился. По определенному полю в этой таблице можно определить, с каким результатом он завершился. В зависимости от

того, удалось ли получить доступ к системе или нет, эксплойт заносится в один из двух списков и запускается следующий.

В результате, пользователь получает отчет о проведенном тестировании системы. Формируется перечень успешно выполненных эксплойтов и соответствующих им уязвимостей, которые администратору системы необходимо ликвидировать. Также выводится статистика по успешным и неуспешным эксплойтам, сколько всего было протестировано и какие (рисунок 15).

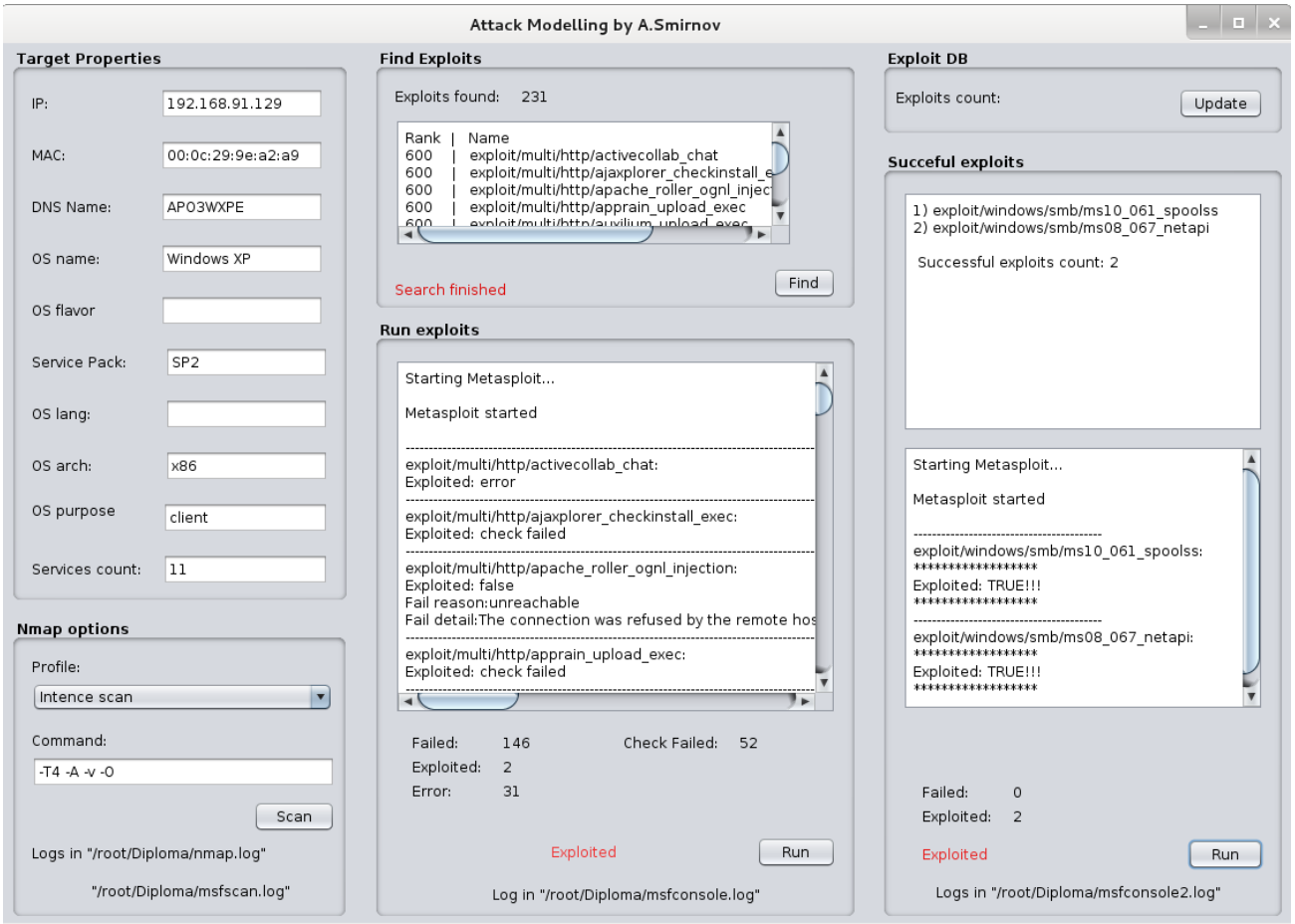

Рисунок 15 – Результаты тестирования системы

5 Тестирование разработанного программного комплекса

С помощью разработанного программного средства было проведено тестирование нескольких операционных систем (Windows, Linux, Mac OS). Часть из них специально предназначена для тестирования новых средств безопасности, другие же являются обычными системами. Результаты приведены в таблице 2.

Таблица 2 Результаты анализа различных ОС

Операционные системы	Количество обнаруженных запущенных служб и открытых портов	Количество отобранных эксплойтов для целевой системы	Количество успешных эксплойтов	Время выполнения
Windows 8.1	11/11	45/1450	1/1	3 мин 02сек 3,75 сек/экспл
Windows 7	10/10	39/1450	3/3	2 мин 30 сек 3,84 сек/экспл
Windows Vista	9/9	33/1450	2/3	3 мин 06 сек 5,6 сек/экспл
Windows XP SP3	9/9	231/1450	1/1	20 мин 02 сек 5,19 сек/экспл
Windows XP SP2	11/11	286/1450	2/2	20 мин 01 сек 4,18 сек/экспл
Windows Server 2008 R2	10/10	37/1450	2/2	2 мин 46 сек 3,98 сек/экспл
Windows Server 2003	4/4	95/1450	3/4	5 мин 30 сек 3, 45 сек/экспл
Mac OS X Mavericks	4/4	37/1450	1/1	5 мин 30 сек 3,75 сек/экспл
Mac OS X Mountain Lion	5/5	65/1450	1/1	4 мин 27 сек 4,23 сек/экспл
Linux Mint 17	2/2	84/1450	0/0	4 мин 34 сек 3,25 сек/экспл
CentOS 7	1/1	8/1450	1/1	0 мин 41 сек 3,17 сек/экспл
Metasploitable 2	30/30	283/1450	7/7	25 мин 45 сек 5,43 сек/экспл
UltimateLAMP	1/1	186/1450	2/2	15 мин 01 сек 4,87 сек/экспл

DVL	2/2	84/1450	8/8	4 мин 32 сек 3,24 сек/экспл

После установки средств защиты ни одному из ранее успешно выполненных эксплойтов не удалось получить доступ к системе. Это связано с тем, что все запускаемые эксплойты общедоступны, а, следовательно, актуальные средства защиты имеют их сигнатуры в своих БД. Однако, существуют методы обхода средств защиты: во-первых, кодирование полезной нагрузки, во-вторых, изменение сигнатуры самого эксплойта.

Поэтому разработанный прототип позволяет загружать и использовать собственные (сторонние) эксплойты, которые, могут не блокироваться защитными средствами.

Для того чтобы оценить эффективность разработанного программного комплекса было проведено сравнение с продуктом Armitage, который имеет схожую функциональность.

Результаты проверки ОС Windows XP SP2 и Metasploitable 2 двумя программами на наличие уязвимостей приведены в таблицах 3 и 4.

Таблица 3 Результаты тестирования ОС Windows XP

Параметры сравнения	Armitage	Разработанный комплекс
Количество обнаруженных запущенных служб и открытых портов	11/11	11/11
Количество отобранных эксплойтов для тестирования	608	286
Количество успешных эксплойтов	2/2	2/2
Время выполнения	28 мин 41 сек 2,8 сек/экспл	20 мин 01 сек 4,18 сек/экспл

Таблица 4 Результаты тестирования ОС Metasplitable 2

Параметры сравнения	Armitage	Разработанный комплекс
Количество обнаруженных запущенных	30/30	30/30

служб и открытых портов		
Количество отобранных эксплойтов для тестирования	353	283
Количество успешных эксплойтов	5/7	7/7
Время выполнения	19 мин 24 сек 3,27 сек/экспл	25 мин 45 сек 5,43 сек/экспл

Помимо Metasploitable 2 и Windows сравнения проводились и на других ОС и количество найденных разработанным прототипом эксплойтов было не меньше, чем у Armitage.

ЗАКЛЮЧЕНИЕ

В результате данной работы были проанализированы существующие методы тестирования средств защиты информации, выбраны вспомогательные инструменты, с использованием которых была разработана архитектура системы моделирования атак и реализован прототип программного средства, который не уступает существующим аналогам.

Разработанную систему можно усовершенствовать в нескольких направлениях:

- возможность аудита целой сети, а не только одного хоста;
- комбинирование нескольких сетевых сканеров;
- автоматизация действий после успешной эксплуатации системы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Kaspersky security bulletin 2014. URL: https://securelist.ru/files/2014/12/Kaspersky-Security-Bulletin-2014-RU.pdf (дата обращения: 09.03.2015).

2 ГОСТ Р 50922-2006 Защита информации. Основные термины и определения. URL: http://vsegost.com/Catalog/57/5737.shtml (дата обращения: 23.03.2015).

3 Common Vulnerability Scoring System, V3 Development Update URL: https://www.first.org/cvss (дата обращения: 12.03.2015).

4 Угрозы, уязвимости и атаки в сетях. URL: http://asher.ru/security/book/its/24 (дата обращения: 21.03.2015).

5 Систематика уязвимостей и дефектов безопасности программных ресурсов. URL: http://www.npo-echelon.ru/doc/is_taxonomy.pdf (дата обращения: 01.04.2015).

6 CVE. URL: https://cve.mitre.org/ (дата обращения: 10.04.2015).

7 Stack, pointers and memory, Lally Singh URL: http://www.biglal.net/Memory.html (дата обращения: 10.04.2015).

8 Metasploit Framework. Прометей эксплойтирования. URL: http://www.securitylab.ru/analytics/216366.php/ (дата обращения: 13.04.2015).

9 Buffer overflows likely to be around for another decade. URL: http://searchsecurity.techtarget.com/news/860185/Buffer-overflows-likely-to-be-around-for-another-decade (дата обращения: 22.04.2015).

10 Buffer overflaws demystified. URL: http://www.enderunix.org/docs/eng/bof-eng.txt (дата обращения: 01.05.2015).

11 How to exploit program vulnerabilities. URL: http://community.core-sdi.com/~juliano/bufo.html (дата обращения: 11.04.2015).

12 Once upon a free(). URL: http://phrack.org/issues/57/9.html (дата обращения: 17.04.2015).

13 Designing shellcode demystified. URL: http://www.enderunix.org/docs/en/sc-en.txt (дата обращения: 17.04.2015).

14 Core Impact Pro. Comprehensive multi-vector penetration testing. URL: http://www.coresecurity.com/core-impact-pro (дата обращения: 19.04.2015).

15 CANVAS, D2 and Tenable Nessus Professional Feed Bundle. URL: http://www.immunityinc.com/products/canvas/d2-nessus-bundle.html (дата обращения: 19.04.2015).

16 SAINT. Vulnerability management, penetration testing, configuration assessment and compliance. URL: http://www.saintcorporation.com (дата обращения: 19.04.2015).

17 Nmap network scanning. URL: http://nmap.org/book/man-output.html (дата обращения: 25.03.2015).

18 Preparing Metapsloit for port scanning. URL: http://www.offensive-security.com/metasploit-unleashed/Port_Scanning (дата обращения: 03.05.2015).

19 Intro to host scanning with NMAP, AMAP, HPING3, XPROBE2, TCPDUMP. URL: http://www.securitytube.net/video/4008 (дата обращения: 07.05.2015).

20 Nmap preset scans. Options and scan types explained. URL: http://www.securesolutions.no/zenmap-preset-scans/ (дата обращения: 21.04.2015).

21 Nmap techniques for avoiding firewalls. URL: https://pentestlab.wordpress.com/2012/04/02/nmap-techniques-for-avoiding-firewalls/ (дата обращения: 09.04.2015).

22 Port scanning techniques. URL: http://nmap.org/book/man-port-scanning-techniques.html (дата обращения: 09.04.2015).

23 Port scanning with Nmap. URL: http://my.safaribooksonline.com/book/networking/security/9781593272883/3dot-intelligence-gathering/active_information_gathering (дата обращения: 14.04.2015).

24 Metasploit. The penetration tester’s guide. URL: http://dieiskandar.blogspot.co.uk/2014/03/metasploit-metasploit-penetration.html (дата обращения: 19.04.2015).

25 Service name and transport protocol port number registry. URL: http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml (дата обращения: 19.04.2015).

26 Understanding and customizing Nmap data files. URL: http://nmap.org/book/nmap-services.html (дата обращения: 17.04.2015).

27 Metasploit exploits stance. URL: http://www.rubydoc.info/gems/metasploitmodel/0.25.1/Metasploit/Model/Module/Stance (дата обращения: 11.04.2015).

28 Meterpreter в деле: хитрые приемы через MSF. URL: https://xakep.ru/2011/03/22/54887/ (дата обращения: 02.05.2015).

29 Chroot, практика. URL: http://linuxru.org/tips/173 (дата обращения: 02.05.2015).

30 Metasploit, MeterpreterClient. URL: http://en.wikibooks.org/wiki/Metasploit/MeterpreterClient (дата обращения: 30.04.2015).

31 Meterpreter basic commands. URL: http://www.offensive-security.com/metasploit-unleashed/Meterpreter_Basics (дата обращения: 29.04.2015).

32 Kali Linux. Penetration testing. URL: https://www.kali.org/ (дата обращения: 30.05.2015).

33 Metasploit Framework (Часть вторая из трех). Прометей эксплойтирования. URL: http://www.securitylab.ru/analytics/216366.php/#ref9 (дата обращения: 20.04.2015).

34 Exploit ranking. URL: https://github.com/rapid7/metasploit-framework/wiki/Exploit-Ranking (дата обращения: 08.05.2015).

35 Class ProcessBuilder. URL: https://docs.oracle.com/javase/7/docs/api/java/lang/ProcessBuilder.html (дата обращения: 03.03.2015).

36 Metasploit updates and msfupdate. URL: https://community.rapid7.com/community/metasploit/blog/2013/01/17/metasploit-updates-and-msfupdate (дата обращения: 05.05.2015).

37 PostgreSQL, JDBC driver. URL: https://jdbc.postgresql.org/documentation/head/intro.html (дата обращения: 02.03.2015).

38 Буферизируемые символьные потоки. BufferedReader и BufferedWriter. URL: http://metanit.com/java/tutorial/6.9.php (дата обращения: 05.04.2015).

39 Metasploitable 2: exploitability guide. URL: https://community.rapid7.com/docs/DOC-1875 (дата обращения: 24.05.2015).